GFA-CLUB Nachrichten

Nr. 3 · 90 1,50 DM

GFA Systemtechnik auf der CeBIT '90



Impressum

Die GFA-CLUB-Nachrichten erscheinen zweimonatlich. Einzelverkaufspreis: 1.50 DM. Der Bezug für GFA-CLUB-Mitglieder ist kostenfrei.

Herausgeber

GFA Systemtechnik GmbH
Heerdter Sandberg 30
4000 Düsseldorf 11

Verantwortlich im Sinne des Presserechts

Joachim Eckstein

Autoren

Ralf Kayser
Hans-Jürgen Merkel
Eduard Rhode
Günter Schmitz
Karl-Heinz Schulze
Andreas Walthes
Martin Wollny

Auf dem Titelfoto

CeBIT '90

Gestaltung der Titelseite

Astrid Theus

Redaktionelle Mitarbeit

Christian Gall
Iris Kayser
Ralf Kayser
Matthias Krejci

Formatierung

Astrid Theus

Druck

Graf & Pflügge

GFA-CLUB-Redaktion
Telefon: 0211/ 55 04-22

Inhaltsverzeichnis

***** Endlich fertig! *****

Ab Ende Mai wird der ASSEMBLER V 1.5 für den Atari ST ausgeliefert.

Das neueste von GFA

Wie in jedem Jahr war GFA auch in diesem Jahr wieder auf der CeBIT vertreten. Da nicht alle Kunden zu unseren Ständen vordringen konnten, möchten wir Ihnen heute unsere Neuheiten und Highlights in aller Ruhe vorstellen.

Für Zeichnungen und Konstruktionen im technischen Bereich hat sich GFA-DRAFT-plus, wie bekannt, bestens bewährt. Nach der PC-Version ist nun auch die Atari-Version um einige nützliche Features erweitert worden. Neben neuen Hilfsprogrammen zur Konvertierung der CAD/CAM-Zeichnungen für das DXF-Format haben Sie nun auch die SPLINES zur Verfügung. Diese Freiformkurven-Funktion ermöglicht nun auch die Konstruktion von komplizierten Bauteilen wie Tragflächen, Strömungslinien oder Kunststoffspritzformen auf einfachste Art. Neu ist auch der Metafiletreiber, über den GFA-DRAFT-Zeichnungen in GEM unterstützende DTP-Programme übernommen werden können. Natürlich ist durch diesen Treiber auch die Ausgabe der Zeichnungen auf dem Atari-Laserprinter möglich. GFA-DRAFT-plus 3.1 für den Atari ist für 398,- DM erhältlich (unverbindl. Richtpreis).

Peter Holzwarth, Roland Schütz und Thomas Ströter hatten sichtlich Probleme mit dem Ansturm auf den GFA-ASSEMBLER 1.5, der auf der CeBIT erstmals vorgestellt wurde. Assemblerfreunde können sich freuen, denn mit dem GFA-ASSEMBLER 1.5 ist das Erstellen von Assemblerprogrammen noch einfacher und schneller geworden. Die Makrofunktionen des Assemblers, die auch schon bisher sehr zur Vereinfachung beitrugen, wurden um Befehle zur Editor- und Ablaufsteuerung erweitert und gestatten nun auch die Verknüpfung von Menuepunkten des Assemblers. Durch diese stark erweiterten Makrofunktionen kann der übliche Arbeitsaufwand bei der Programmerstellung sicherlich um 20% gesenkt werden. Ein übriges tut eine neue Option, die den Zehnertastaturblock in Textsteuerungstasten umwandelt. GFA-ASSEMBLER für den Atari und Amiga kosten je 149,- DM (unverbindl. Richtpreis). Darüber hinaus versprechen wir, in der nächsten Ausgabe einen umfassenden Test zu bringen.

Auch Dirk Van Assche und Lars van Straelen hatten ihre liebe Mühe auf der Messe, denn das neue GFA-BASIC 3.5 mit den vielen Zusatzprogrammen zog die User wie ein Magnet an. Ein besonderer Leckerbissen ist wohl die neue GFA-BASIC Grafik- & Sound-Bibliothek, denn wer möchte seine Programme nicht mit einem an Hollywood erinnernden Outfit versehen? Interruptanimation, Überblendungen, Shapes in verschiedenen Größen, Digitaler Sound oder gar Vertikal- und Horizontalscrolling ohne Programmieraufwand sprechen für sich und stellen doch nur einen kleinen Teil der Bibliothek dar. Die Grafik- & Sound-Bibliothek ST kostet mit einem 249seitigen Handbuch 149,- DM (unverbindl. Richtpreis).

Für alle Arten des GEM-Handling unter GFA-BASIC ist unser GFA-GUP genau richtig. Menuehandling, Resourceaktionen und Fensterverwaltung mit bis zu sieben Fenstern stellen jetzt auch für Einsteiger kein Problem mehr dar. GFA-GEM-Utility Package kostet mit Handbuch 149,- DM.

Um GFA-BASIC-Programme vom ST auf andere Computertypen wie MAC oder IBM zu übersetzen, war es bislang nötig, das Programm in einen anderen, meist weniger komfortablen Dialekt umzuschreiben. Mit unserem GFA-BASIC-Konverter nach C können die Basic-Programme nun in die C-Sprache umgewandelt werden und so auch auf anderen Rechnertypen laufen. Durch die Verwendung von GFA-BASIC bei der Programmerstellung verbinden Sie die kurzen Turn Around Zeiten (die Zeiten zum Testen und Fehlerbeheben) mit der benutzerfreundlichen Bedienungsoberfläche und einfachen Handhabung. Der GFA-BASIC-Konverter nach C ist für 498,- DM (unverbindl. Richtpreis) zu haben und erzeugt einen sofort verarbeitungsfähigen Standard-C-Quelltext.

Auf dem Atari hat die strukturierte Programmierung seit dem Erscheinen des GFA-BASIC immer mehr zugenommen. Für viele Programmierer türmen sich bei der Strukturerstellung von Programmen jedoch große Probleme auf, da sich das Zusammenspiel einzelner Strukturteile oder Routinen im Programmlauf sehr schwer erfassen läßt. Meist scheitern Softwareprojekte nicht an der Programmiererfahrung oder Kenntnis, sondern an endlosen Fehlersuchen und Optimierungen. Durch GFA-STRUKTO, dem Struktur-Editor, der nebenbei auch als Texteditor Verwendung findet, können Sie nun einfach und schnell Programme beliebiger Art strukturieren und optimieren. Selbst die Fehlerbeseitigung - oftmals der größte Arbeitsaufwand - kann nun erheblich verkürzt werden, da auf einen Blick klar wird, wann wo was abläuft. GFA-STRUKTO kostet mit umfangreichem Handbuch und drei Beispielprogrammen für Grafik, Multitasking und Simulation 249,- DM.

Filmähnliche Animationen werden durch das GFA-ZOETROPE Animations-System auf dem Amiga zu einem Kinderspiel. Bewegungspläne, Cel-Animationen, Zooms, Vergrößerungen und Verkleinerungen gestatten die Erstellung von professionellen Videofilm-Einführungen oder aufwendigen Grafikdemos ohne große Programmierkenntnis, da ZOETROPE alle notwendigen Funktionen auf Ihr Mauskommando hin aktiviert. GFA-ZOETROPE wird mit einem Grafikprogramm, dem Animator, zwei Konvertern, vielen Beispielen und dem umfangreichen Handbuch für 198,- DM ausgeliefert (unverbindl. Richtpreis).

Das Erstellen von Statistiken war schon immer recht aufwendig. Einzelne Berechnungsarten und Vergleichsmöglichkeiten gestalteten sich meist so umfangreich, daß eine einwandfreie Statistik nur auf Großrechnern möglich wurde. Durch GFA-STATISTIK können nun auch auf dem ST Inferenzstatistiken, Parametrische und Nichtparametrische Tests, deskriptive Statistiken und komplexere Verfahren wie uni- und multivariate Tests durchgeführt werden. Selbst die Datenmanipulation von fehlerhaften Werten kann durch Vertauschen, Ersetzen und Einfügen leicht abgeschlossen werden, so daß nach kurzer Zeit ausgewertete Statistiken mit Grafik zur Verfügung stehen. GFA-STATISTIK kostet in der Vollversion 998,- DM, in der Studentenversion 398,- DM (unverbindl. Richtpreis) und wurde von Mitarbeitern der Uni Düsseldorf entwickelt.

Sparringspartner für die oberste Etage

Die Welt ist hektischer geworden, die Anforderungen an den Einzelnen sind rapide angewachsen und die "menschliche" Hilfe ist in vielen Bereichen nicht mehr so gefragt wie früher. An vielen Orten hat der Computer Einzug gehalten, so auch in den Bereichen Ausbildung und Lernen. <VOKAV plus>, ein Trainingspartner beim Erlernen von Fremdsprachen, verspricht "DAS intelligente Vokabelsystem für alle Atari ST" zu sein. Hm ja, ob mit intelligent der Atari ST, das Programm oder der Benutzer gemeint ist... ?

Das Programm würde ich zwar bestimmt nicht als intelligent, aber doch als recht gut und vor allem als anwenderfreundlich bezeichnen. Bei Unklarheiten mit der Bedienung hilft ein kleines Handbuch im DIN A5 - Format über die ärgsten Hürden locker hinweg. Es ist klar, übersichtlich und ohne Firlefanz gegliedert und beschreibt auch dem weniger Geübten, wie er mit dem Programm umgehen kann. Positiv sind dabei vor allem die einzelnen Anwendungsbeispiele mit Abbildungen der dazugehörigen Menuboxen.

Für mich persönlich sollte ein Handbuch auch nur vereinzelt zum Nachsehen Verwendung finden, wobei der Frust meist im Quadrat mit der Seitenzahl, bzw. mit vielem Blättern und Wälzen in "Kraut- und Rübenfeldern", wächst. Frust gab's hier eigentlich überhaupt keinen, auch wenn man manche Abschnitte vielleicht überarbeiten könnte.

So einleuchtend das Handbuch, so einfach ist eigentlich auch die Bedienung des Programms (ich hab's auch gleich begriffen...). Eine Lerndatei, die man bei entsprechendem Umfang durchaus als Fremdwörterlexikon verwenden kann, besteht aus Wörtern / Begriffen einer fremden, und gegenübergestellt, der deutschen Sprache. Selbstverständlich könnten es auch zwei Fremdsprachen sein, deren Begriffe sich paarweise gegenüberstehen. Normalerweise stehen pro Begriffspaar 2 x 38 Zeichen zu Verfügung, was ausreichen dürfte. Im Moment verarbeitet das Programm laut Handbuch bis zu 6000 Paare pro Vokabeldatei, wobei sich nur eine davon im Speicher befinden kann. Vielleicht werden es ja in zukünftigen Updates noch mehr...

In der Menuleiste des Programms findet man nach dem Info und den nicht aufrufbaren Acc's den Punkt <DISK>. Hier lassen sich Vokabeldateien einlesen bzw. neu anlegen, normal und als Textdatei zur Weiterverarbeitung in anderen Programmen abspeichern.

Auch die Bearbeitung von Blöcken ist zu finden, und außer dem Beenden des Programms noch einige Diskettenutilities.

Unter <VOKABELN> geht's jetzt in die Vollen. Unter 'Vokabeln eingeben' wird in einer übersichtlichen Box zuerst nach der fremden, dann nach der deutschen Vokabel gefragt. Hier kann auch ein Schwierigkeitsgrad gewählt werden. Wichtig dabei ist bei Mehrfachbegriffen eine Trennung mit <; oder />, da es sonst Probleme beim Suchen von Begriffen gibt.

Suchen kann man nach Vokabeln und per Jokerzeichen auch nach Teilen von ihnen. Eine Schwierigkeit tauchte dabei trotz obigem Hinweis auf. Ein Beispiel aus einer Lerndatei: Fremde Vokabel: to bear, bore, born, deutsche Vokabel: ... Sucht man jetzt nach 'bear', so findet das Programm nichts! (zumindest bei mir; vielleicht hängt das mit der Intelligenz zusammen...). Spaß beiseite; ein kleiner Trick hat mir weitergeholfen. Stellt man einen '*' als Joker nach, dann funktioniert es prima. Vielleicht kann dieses kleine Manko noch ausgemerzt werden.

Beinahe hätte ich's vergessen. Suchen kann man selbstverständlich nach den verschiedensten Kriterien, wie z.B. Deutsch/Fremd, Block/Alle oder Groß/Klein, also mit oder ohne Unterscheidung von Großbuchstaben. Bei Bedarf lassen sich die Suchbegriffe auch durch neue Begriffe ersetzen. Auch Sortieren kann man die Vokabeln auf verschiedene Arten.

Ein weiterer umfangreicher Menupunkt beschäftigt sich mit dem Ausdruck. Hierbei können nicht nur rechter und linker Rand, Druckqualität und Schrifttypen ausgewählt werden, sondern auch Kopf- und Fußzeilen und Menge. So ziemlich jeder Drucker dürfte optimal bedient werden, da VOKAV plus mit jedem 1st Word plus Treiber zusammenarbeitet, der nur in 'printer.cfg' umbenannt werden muß.

Noch eine zentrale Aufgabe des Programms ist hier zu finden. Was wäre ein Vokabelprogramm ohne Lernfunktion? Richtig, natürlich nichts! Die Auswahlbox bietet schon einiges. Festlegen kann man die Anzahl, den Schwierigkeitsgrad, und ob bereits eine Bedeutung ausreicht, oder alle angegebenen Begriffe eingegeben werden müssen. Abgefragt wird auch, ob die Reihenfolge auf- oder absteigend oder zufällig, die Vorgabe in deutsch, fremd oder zufällig ist, ob neugelernt oder überlernt wird. Nicht vergessen darf man, daß das Programm auch verschiedene Kürzel kennt und natürlich am Schluß eine Übersicht über den Lernerfolg ausgibt. Der kann ganz frustrierend sein... Puh, das reicht aber jetzt wirklich damit...

Unter <BLOCK> sind gängige Blockfunktionen wie Löschen, Kopieren, Verschieben und Ändern aufrufbar. Hilfestellungen gibt's auch hier durch prima Auswahlboxen. Als letzter Menupunkt wäre da noch <EXTRAS>. Unter 'Information' wird freier Speicher etc. angezeigt. Außerdem läßt sich hier der Druckertreiber installieren. Eine ordentliche Funktionstastenbelegung erlaubt Arbeitserleichterungen bei oft benötigten oder auch außergewöhnlichen Begriffen. Zuletzt lassen sich alle Einstellungen abspeichern und bei Bedarf neu einlesen.

Zusammenfassend läßt sich sagen, daß VOKAV plus 2.0 ein angenehm zu bedienendes, übersichtliches und praktisches Programm ist. All jenen, die in irgendeiner Form mit dem Vokabellernen konfrontiert sind, kann man dieses Programm wirklich empfehlen. Es dient bestimmt auch prima als Nachschlagewerk. Angesprochen sind hier nicht nur Privatpersonen, sondern vor allem auch die entsprechenden Schulen, die im Fremdsprachenunterricht damit ohne Einarbeitungszeit für Schüler und Lehrer umgehen können.

Interessant ist in diesem Zusammenhang bestimmt, daß ein auch als Accessory lauffähiger Lexikonzusatz mitgeliefert wird. Ebenfalls auf Diskette befinden sich ein Konvertierprogramm für ältere Dateien und ein Programm zum Teilen bzw. Zusammenfügen von Vokabeldateien. Betrachtet man außerdem den wirklich günstigen Preis und die Tatsache, daß es für ca. 20,- DM je eine Vokabeldatei für französisch, spanisch, italienisch und englisch zu erwerben gibt, kann man bestimmt von einer 'runden Sache' im Sinne der Anwender sprechen. Nicht verschweigen möchte ich noch, daß das Programm ohne Schwierigkeiten auf meiner Festplatte läuft...

(wh)

Progr.-Name:	Vokav plus 2.0
Kopierschutz:	nein
Auflösungen:	monochrome
Handbuch:	DIN A5, deutsch, übersichtlich
Preise:	Programm: 59,- DM Vers. Campus: 39,- DM Wortschatz: je 19,- DM
Copyright:	Arnd von Wedemeyer Mettmanner Str. 12 5603 Wülfrath

112.640 Bytes zum Nulltarif

Der Atari ST formatiert Disketten standardmäßig mit 80 Tracks zu je 9 Sektoren. Da jeder Sektor 512 Bytes groß ist, macht das bei einer doppelseitigen Diskette 737.280 Bytes (80x9x512x2). Nach Abzug etlicher Kilobytes für die Diskettenverwaltung bleiben 720 KB zur freien Verfügung.

Bei der Formatierung geht das Betriebssystem recht verschwenderisch mit dem Platz auf der Diskette um und kalkuliert eine üppige Sicherheitsreserve ein. Tatsächlich ist es ohne Probleme möglich, auf jeder Spur einen 10. Sektor unterzubringen. Allein durch diese Maßnahme gewinnen wir auf einer doppelseitigen Diskette zusätzliche 81.920 Bytes Speicherplatz.

Es kommt noch besser: Wir müssen uns keineswegs mit 80 Tracks begnügen; die meisten Laufwerke vertragen mehr, manche sogar 86. Als vernünftiger Kompromiß hat sich in eigenen Versuchen ein Wert von 83 erwiesen. Das ergibt weitere 30.720 Bytes. Macht zusammen 112.640 Bytes, die keinen Pfennig Geld kosten.

Mehr als 83 Tracks können problematisch werden. Die Grenzen Ihres eigenen Laufwerks müssen Sie durch Ausprobieren herausfinden. Aber Vorsicht: Sie sollten hierbei nicht nur die Laufwerksmechanik im Auge (bzw. im Ohr) behalten, sondern auch der Datensicherheit die nötige Aufmerksamkeit schenken. 85 Tracks z.B. nutzen wenig, wenn Sie ständig von Schreib-/Lesefehlern geplagt werden.

Noch ein Hinweis: Mit FORMAT.PRG formatierte Disketten können Sie nicht mehr als Ganzes, sondern nur noch dateiweise kopieren, sofern Sie nicht eines der Copyprogramme verwenden. Das gilt für jede Diskette, die vom ST-Standardformat abweicht.

Günter Schmitz

```
; FORMAT.PRG © 1989 by Günter Schmitz
;               Bruchstr. 167
;               5024 Pulheim
;               Tel. 02238/14877
;
; GFA-ASSEMBLER V1.3
;
; Files: FORMAT.IS
;        FORMAT.PRG
;
; I: 25.09.1989
; U:
;
; Formatiert Disketten mit 83 Tracks zu jeweils 10
; Sektoren. Dadurch erhöht sich die Kapazität
; einer doppelseitigen Diskette um 112.640 Bytes
; und die einer einseitigen Diskette um 56.320 Bytes.
;
; 83 Tracks sind ein Kompromiß. Manche Laufwerke
; vertragen mehr. Um es auszuprobieren, muß man die
; entsprechenden Stellen im Listing ändern.
;
gemdos    = 1
cconin    = 1     ; Zeicheneingabe mit Echo
cconout   = 2     ; Zeichenausgabe
crawcin   = 7     ; Zeicheneingabe ohne Echo
cconws    = 9     ; String-Ausgabe
;
xbios     = 14
flopfmt   = 10    ; Diskette formatieren
flopwr    = 9     ; Sektoren schreiben
protobt   = 18    ; Boot-Sektor erzeugen
;
; Startmeldung und Eingabeaufforderung
          lea.l    cls(pc),a0     ; Bildschirm löschen und
          bsr      prline         ; Startmeldung ausgeben
          move.w   #crawcin,-(sp) ; auf Taste warten
          trap     #gemdos
          addq.l   #2,sp

eingabe:  lea.l    wahl(pc),a0    ; Angabe, ob 1- oder
          bsr      prline         ; 2seitiges Laufwerk
          move.w   #cconin,-(sp)  ; auf Eingabe warten
          trap     #gemdos
          addq.l   #2,sp
          subi.b   #49,d0         ; 1seitiges Laufwerk?
          beq.s    format
          cmpi.b   #1,d0          ; 2seitiges Laufwerk?
          bne.s    eingabe        ; wenn nein -> Wiederholung
;
; Die gesamte Diskette wird mit $e5 formatiert
format:   move.w   d0,d6          ; Seitenzahl retten
          move.w   #82,d3         ; 83 Tracks (0 .. 82)
next:     move.w   d6,d4
repeat:   lea.l    anzeige(pc),a0 ; Anzeige des gerade in
          bsr      prline         ; Arbeit befindlichen Tracks
          move.l   d3,d5
          divu.w   #10,d5         ; Zehner-Stelle
          addi.w   #48,d5
          move.w   d5,-(sp)
          move.w   #cconout,-(sp)
          trap     #gemdos
          addq.l   #4,sp
          swap.w   d5
          addi.w   #48,d5         ; Einer-Stelle
          move.w   d5,-(sp)
          move.w   #cconout,-(sp)
          trap     #gemdos
          addq.l   #4,sp
          move.w   #$e5e5,-(sp)   ; Initialdaten
          move.l   #$87654321,-(sp) ; Magic number
          move.w   #1,-(sp)       ; keine Sektoren überspringen
          move.w   d4,-(sp)       ; Seite
          move.w   d3,-(sp)       ; Track
          move.w   #10,-(sp)      ; 10 Sektoren pro Track
          clr.w    -(sp)          ; Laufwerk A
          pea      0
          pea      puffer(pc)     ; Track-Puffer
          move.w   #flopfmt,-(sp)
```

```
        trap    #xbios
        lea.l   26(sp),sp
        dbra    d4,repeat       ; nächste Seite
        dbra    d3,next         ; nächster Track
;
; Die ersten beiden Tracks ausnullen
        move.w  d6,d4           ; Seitenzahl nach d4
seite:  moveq.l #1,d3           ; Tracks 0 und 1
null:   clr.w   -(sp)
        move.l  #$87654321,-(sp) ; Magic number
        move.w  #1,-(sp)        ; keine Sektoren überspringen
        move.w  d4,-(sp)        ; Seite
        move.w  d3,-(sp)        ; Track
        move.w  #10,-(sp)       ; 10 Sektoren pro Track
        clr.w   -(sp)           ; Laufwerk A
        pea     0
        pea     puffer(pc)      ; Track-Puffer
        move.w  #flopfmt,-(sp)
        trap    #xbios
        lea.l   26(sp),sp
        tst.w   d0
        bmi.s   fehler          ; das ging schief
        dbra    d3,null         ; nächster Track
        dbra    d4,seite        ; nächste Seite
;
;Boot-Sektor aufbauen
        clr.w   -(sp)           ; Boot-Sektor nicht ausführbar
        addq.w  #2,d6           ; d6 = 2 : 1seitiges Laufwerk
        move.w  d6,-(sp)        ; d6 = 3 : 2seitiges Laufwerk
        move.l  #$1000000,-(sp) ; Seriennummer
        pea     puffer(pc)
        move.w  #protobt,-(sp)
        trap    #xbios
        lea.l   14(sp),sp
        move.b  #10,puffer+24   ; 10 Sektoren pro Track
        subq.w  #1,d6
        move.l  #830,d0         ; 83 Tracks * 10 Sektoren
        mulu.w  d6,d0           ; * Seitenzahl = Gesamtzahl
        divu.w  #256,d0         ; wird im Intelformat abge-
        move.b  d0,puffer+20    ; legt
        swap.w  d0
        move.b  d0,puffer+19
;
; Bootsektor schreiben
        move.w  #1,-(sp)        ; ein Sektor
        pea     0               ; Seite 0, Track 0
        move.w  #1,-(sp)        ; Sektor 1
        clr.w   -(sp)           ; Laufwerk A
        pea     0
        pea     puffer(pc)
        move.w  #flopwr,-(sp)
        trap    #xbios
        lea.l   20(sp),sp
        bra.s   exit
;
; Fehlermeldung
fehler: lea.l   err_msg,a0
        bsr     prline
        move.w  #crawcin,-(sp)
        trap    #gemdos
        addq.l  #2,sp
;
; Programmende
exit:   clr.w   -(sp)
        trap    #gemdos
;
; Text ausgeben
prline:  move.l  a0,-(sp)
         move.w  #cconws,-(sp)
         trap    #gemdos
         addq.l  #6,sp
         rts
;
; Meldungstexte
         .DATA
cls:     .DC.b 27,"E","Liegt die richtige Diskette in
               Laufwerk A ?",13,10
         .DC.b "Dann bitte irgendeine Taste drücken.",13,10,0
wahl:    .DC.b 27,"Y",5+32,32
         .DC.b 27,"p 1",27,"q- oder ",27,"p 2",27,
               "qseitige Diskette? ",0
anzeige: .DC.b 27,"Y",7+32,32,"Track .. ",0
err_msg: .DC.b 13,10,13,10,7,"Fehler ist aufgetreten. "
         .DC.b "Bitte Taste drücken.",0
         .EVEN
         .BSS
puffer:  .DS.b 8192
```

GFA-BASIC und GFA-ASSEMBLER

Immer wieder werde ich über die Helpline gefragt, wann denn der Einsatz von GFA-BASIC oder GFA-ASSEMBLER sinnvoll ist. Obwohl zu diesem Thema ein empfehlenswertes Buch von GFA erschienen ist, möchte ich an dieser Stelle einmal für alle die Grundkriterien einer solchen Entscheidung darlegen.

Da das GFA-BASIC 3.5 in seiner derzeitigen Version für fast alle Anwendungen geeignet ist, bietet es sich geradezu an, neue Ideen oder Vorhaben schnell in die Tat umzusetzen. Durch die Compiler/Interpreter-Paarung können so lauffähige und voll funktionstüchtige Programme erstellt werden, die in Punkto Geschwindigkeit nicht versteckt werden müssen. Tatsächlich ist der größte Vorteil des Basic die Option, Programme und Routinen sofort ohne Assemblier- und Linkzeit auszutesten, um so eventuell vorhandene Fehler auszumerzen. Auch ist die Basic-Sprache im Gegensatz zur Maschinen-Sprache leicht zu erlernen und schnell anzuwenden.

Ein Manko der von Compilern hergestellten Programme ist deren Länge. Kein Compilat kann so kompakt und schnell sein wie ein komplett oder auch nur teilweise in Assembler geschriebenes Programm. Dies liegt jedoch im Wirkungsprinzip und gilt ebenfalls für die übrigen Compilersprachen wie 'C', 'PASCAL' oder gar 'MODULA2'.

Nie kann ein Compiler so flexibel auf die individuellen Bedürfnisse des Autors eingehen wie der Programmierer, da nur er genau abschätzen kann, welche Anforderungen an eine Programm-Routine gestellt werden. Bei einer einfachen Rechnung innerhalb eines Assemblerprogramms wird zum Beispiel nur für diese Rechnung eine Routine geschrieben, die dann auch nicht zu anderen Berechnungen passen muß. Beim Compiler und Interpreter hingegen sind diese Routinen für den Anwender flexibel gehalten und somit etwas langsamer und umfangreicher.

Durch den INLINE-Befehl des GFA-BASIC 3.x Entwicklungssystems in Verbindung mit dem GFA-ASSEMBLER 1.5 kann dieser Nachteil jedoch gut behoben werden. So lassen sich relativ einfach schnelle und kompakte Programme erstellen, ohne auf die Vorzüge des GFA-BASIC 3.5 verzichten zu müssen. Hierdurch ist es auch möglich, ein Programm komplett in Basic zu verfassen, um dann Schritt für Schritt zeitkritische und speicherfressende Routinen durch spezielle und entsprechend schnelle Assemblerroutinen zu ersetzen. Das beste Beispiel hierfür ist die im Buch 'GFA-BASIC Version 3' beschriebene Printroutine, die durch gezielten Einsatz der Einbindungsmöglichkeiten im GFA-BASIC den Printbefehl um den Faktor von 2.4 beschleunigt.

Assembler haben den Nachteil, daß die Programmerstellung sehr mühsam ist. Selbst Grundfunktionen, wie die unformatierte Textausgabe, müssen erst geplant und entwickelt werden, wodurch der Sourcetext zum einen sehr lang und zum anderen zeitaufwendig sein kann. Der CLS-Befehl in Basic ergibt ohne Betriebssystemaufrufe in Assembler schon 10 bis 20 Zeilen reinen Sourcetext.

Bei dem GFA-ASSEMBLER wird jedoch ein Großteil dieses Mankos durch die Verwendung von Makros sowie den Befehlskürzeln wieder wett gemacht. Nebenbei haben die Programmierer auch noch die Möglichkeit der Tastaturprogrammierung gegeben, womit das Erstellen von Programmen ebenfalls stark vereinfacht wird.

Nicht zu verleugnen ist auch, daß alle Assembler ein gewisses Maß an Systemkenntnis voraussetzen und die Assemblersprache nicht einfach zu erlernen ist. Aus diesem Grund haben wir in dieser Ausgabe einen Assembler-Kurs gestartet, der auf dem GFA-BASIC aufbaut und Ihnen die Grundkenntnisse des Assemblers zugängig machen soll.

Fast unverzichtbar ist der Assembler bei Direktzugriffen auf die Hardware. So ist es kaum vorstellbar, die Programmierung des FDC, des Tastaturchips oder gar des Videochips unter Basic zu erledigen. Hierbei stehen eindeutig die Geschwindigkeit und die Kürze der Unterprogramme im Vordergrund. Nebenbei bemerkt, es wäre wünschenswert, wenn Atari beim TOS ebenfalls einen Assembler verwendet hätte, denn dank C ist das (X)BIOS für den Grafikchip etwas zu langsam geraten.

Basic würde ich bei Resourcen den klaren Vortritt lassen, denn im Umgang mit AES und VDI zeigt GFA-BASIC 3.x seine wahre Stärke. Funktionen à la CARD und WORD oder Objektvariablen wie OB_SPEC machen den Umgang mit Resourcen zu einem echten Vergnügen. Nicht zuletzt wirkt sich hier auch die konsequente Anwendung von Pointern und vordefinierten Betriebssystemfunktionen wie RSRC_LOAD stark vereinfachend aus, zumal hier selbst mit dem GFA-ASSEMBLER kein nennenswerter Geschwindigkeits- oder Speichergewinn zu verzeichnen ist und so beruhigt auf GFA-BASIC gesetzt werden kann.

Ein echtes Problem für Assemblereinsteiger sind die mathematischen Funktionen wie Sinus, Cosinus oder Wurzelberechnungen. Diese recht aufwendig zu programmierenden Funktionen sind ebenfalls in Basic gut aufgehoben, sofern eine Tabellenlösung innerhalb des Programmes nicht möglich ist.

Hingegen ist bei Programmteilen für großangelegte Speichermanipulationen wie BITBLK oder das Umwandeln von Bildern einer Auflösung in die andere immer der Assembler vorzuziehen. Die hierdurch erzielte Geschwindigkeit läßt Routinen, die unter BASIC an die 2 min. verbrauchen würden, in unter 10 sec. ablaufen. Nach Murphy kommen solche Routinen ja leider recht oft vor, so daß sich der Assembler dann recht erholsam auswirkt.

Durch den RCALL-Befehl im GFA-BASIC 3.x können nunmehr sogar ganze Daten und Adresslisten in die Register D0-D7 und A0-A6 geladen werden, womit auch die diversen Tricksereien bei der Parameterübergabe hinfällig wurden.

Durchaus denkbar ist es auch, Such-, Save- und Laderoutinen innerhalb einer Adressverwaltung oder Datenbank komplett in Assembler zu verfas-

sen und aus dem Basic heraus nur die zugehörigen Pointer zu übergeben. Die Adressliste V 1.0 des GFA- CLUBS hatte ich komplett in GFA-BASIC 3.x geschrieben, wobei die Adressdatei mit fast 140 Kbytes dank RECALL fast drei Minuten zum Laden benötigte. In der Version 2.0 hatte ich schon die Lade- und Speicherroutine durch den Assembler ersetzt, wodurch das Laden nunmehr nur noch 21 sec. dauerte.

Wegen der Speicherkapazität - das BASIC-Programm hatte 28 Kbytes - habe ich nun die neue Version komplett in Assembler geschrieben und das Programm hierdurch bei größerem Funktionsumfang auf knapp acht Kbytes verkürzt.

Natürlich stellt diese Aufzählung nur einen kleinen Ausschnitt der möglichen Anwendungen dar, eine komplette Aufstellung würde auch den hier gesetzten Rahmen sprengen, so daß ich nur noch mal auf die angesprochenen Bücher hinweisen möchte. Letztlich wird jeder User selbst merken, wann eine Routine besser in Assembler oder Basic verfaßt wird. Als kleinen Anreiz haben wir ab dieser Ausgabe einen Assemblerkurs für Einsteiger, der es Ihnen leichter machen soll, Ihre eigenen Vorstellungen zu verwirklichen.

Literaturliste:

GFA-BASIC ST Version 3.0
ISBN 3-89317-004-9
inkl. Diskette DM 59,-

GFA Anwenderbuch
ISBN 3-89317-001-1
inkl. Diskette DM 59,-

Programmieren in
Maschinensprache
SYBEX Verlag
ISBN 3-88745-512-6
DM 48,-

PC-SPEED

Erfahrungen und Tips zum Hardware-PC/XT-Emulator

Oder ... Emulator, Emulator im ST - bist auch so schnell wie ein XT?

Als Informatiklehrer an einer Gesamtschule bin ich (leider?) in meinem Unterricht gezwungen, auf die vom Schulträger angeschafften ATARI-PCs zurückzugreifen. Vorbereitungen waren eigentlich anfangs nur in Freistunden möglich, da zu Hause ja ein ST seinen Dienst versah. Programmübertragungen waren dadurch ausgeschlossen.

Nachdem dann endlich ein funktionsfähiges Emulationsprogramm in Form von PC-DITTO vorlag, konnte ich doch leicht aufatmen. Trotz der ziemlich langsamen Arbeitsgeschwindigkeit (mit PC-TOOLS gemessene Arbeitsgeschwindigkeit betrug 30% der normalen PC-Geschwindigkeit) war gewährleistet, daß ich Schulprogramme auch zu Hause laufen lassen konnte.

Seit Sommer '89 liegt nun der lang ersehnte Hardware-Emulator PC-SPEED vor. Die vom Vertrieb (HEIM-Verlag in Heidelberg) angekündigten Eigenschaften ließen mich gleich aufhorchen. Nach einem letzte Fragen klärenden Gespräch mit dem Verlag bestellte ich dann den PC-SPEED.

Die Lieferung bestand aus einer nicht allzu großen Platine und einem kleinen 15-seitigen Anleitungsheftchen. Die notwendige Software befand sich auf einer beigefügten Diskette. Trotz der warnenden Hinweise auf Seite 1 der Anleitung bezüglich der Garantie (war eh schon abgelaufen) und des vielleicht nicht so ganz einfachen Einbaus (... wer hat Angst vor dem schwarzen Chip??) begann ich gleich, meinen ST auseinanderzunehmen.

Zunächst deckte ich alle in der Nähe der CPU liegenden Teile sorgfältig mit isolierendem Klebeband ab. Auch die Tastatur wurde von unten mit Isolierband abgeklebt. Die Platine des PC-SPEED liegt zum Teil verdammt nah an einigen Bauteilen der ST-Platine bzw. den Drahtbrücken auf der Rückseite der Tastatur.

Die heißeste (im wahrsten Sinne des Wortes) Phase des Unternehmens stand mir dann gleich bevor: das Einlöten des mitgelieferten CPU-Sockels. Mit einem 30 Watt-Lötkolben klappte dies aber ganz vorzüglich. Wichtig dabei war, daß ich nicht zu lange, mit einigen Sekunden Pause dazwischen, die vielen Beinchen des Sockels auf den 68000er auflötete. Hierbei möchte ich jedem Nachahmer empfehlen, die Lötstellen und die Steckkontakte des Sockels anschließend mit einem Multimeter (mit Durchgangsprüfer) zu testen.

Als nächstes schnitt ich dann mit einer Blechschere ein entsprechend großes Viereck aus der Blechabschirmung heraus, da bei meinem 1040er die CPU direkt ganz vorne liegt und der PC-SPEED anders nicht genügend Platz hat. Dann konnte ich die Platine des Emulators auf den aufgelöteten Sockel aufdrücken. Keine Angst, so schnell geht da nichts kaputt! Wichtig ist, daß alle Kontakte gute und feste Verbindung mit dem Sockel haben. So tief wie möglich eindrücken, damit nachher die Tastatur nicht hochgedrückt wird!

Zum Testen habe ich die Einzelteile nur lose eingebaut (aber mit Gehäusedeckel, wegen 220 Volt Netzspannung!). Das auf der Diskette mitgelieferte Startprogramm PC_SPEED.PRG wurde dann auf seine "BUS"-Reise geschickt. Doch leicht nervös habe ich dem Klackern des ST-Laufwerks gelauscht und fand mich nach einigen Sekunden in der MS-DOS-Welt wieder, wo man mich aufforderte, eine Systemdiskette einzulegen.

Nach insgesamt etwa 2-stündiger Arbeit war es dann soweit: eine MS-DOS-Systemmeldung und das unvermeidliche Promptzeichen " A> " blinkten mich an. Gleich wurden einige MS-DOS-Programme ausprobiert, unter anderem PC-TOOLS. Ich wollte doch wissen, ist "er" wirklich so schnell wie ein XT? Sage und schreibe 250% Arbeitsgeschwindigkeit mel-

dete mir das Programm. Damit läßt es sich arbeiten.

Gespannt war ich auf den Einsatz der Maus. Nach Starten des Maustreibers war ich aber doch leicht enttäuscht. Ein mausunterstütztes Programm ließ sich nicht bedienen. Zwar konnte ich alle Funktionen des Programms über die Tastatur aufrufen, jedoch ein Mauscursor ließ sich nicht blicken. Dieses Problem ist inzwischen behobenworden, und die Original-ATARI-Maus kann in der vorliegenden Version 1.30 benutzt werden. Notwendig ist jedoch ein, bei vielen MS-DOS-Programmen mitgelieferter, MICROSOFT-kompatibler Maustreiber.

Alle meine getesteten Programme (TURBO-BASIC, QUICK-BASIC, GW-BASIC, WORDSTAR, div. PD-Programme und Schulprogramme [Textverarbeitung, Matheprogramme etc.]) liefen zu meiner vollsten Zufriedenheit.

Einzig mit der Wiedergabe von Sound bin ich nicht ganz zufrieden. Leider ist dieser Punkt noch nicht hundertprozentig korrekt gelöst. Bei Spielen vor allem (BATTLE CHESS, FLUG-SIMULATOR etc.) wird nur ein undefinierbares Pfeifen und Piepsen übertragen mit dem Haken, daß das ganze Programm oft zum Teil entsetzlich lange unterbrochen wird. Ich empfehle den Sound abzustellen, da die langen Wartezeiten doch mehr als störend sind. Ich denke, daß Hansjörg Sack dies aber auch noch in den Griff bekommt.

Fazit: Nimmt man einen momentanen Verkaufspreis für einen 1040er ST mit eingebautem PC-SPEED von ca. 1700 DM, so erhält man eine kaum schlagbare Kombination von TOS und MS-DOS, um die uns so mancher "nur" MS-DOS-Benutzer beneiden kann. Die schon erwähnte sehr hohe Arbeitsgeschwindigkeit, der relativ problemlose Einbau, der günstige Preis (...haben Sie noch 2000 bis 3000 DM für einen XT übrig?) und die einwandfreie Funktion lassen eigentlich keine Alternative zu.

Wenn Sie noch Fragen zu PC-SPEED haben, können Sie sich gerne an mich wenden oder Herrn Till vom HEIM-Verlag anrufen (06151/57783). Als sehr hilfreich möchte ich Ihnen das gerade neu erschienene Buch "PC-SPEED" aus dem HEIM-Verlag ans Herz legen, da es viele sehr gute Tips bereithält, die die Arbeit und das Installieren von PC-SPEED erleichtern.

Im Anschluß finden Sie noch ein kleines Programm, das Ihre ST-Disketten auch MS-DOS-lesefähig macht. Da im Prinzip das ST-Format und das DOS-Format gleich sind, lassen sich die ST-formatierten Disketten mit diesem kleinen Patch auch unter PC-SPEED verwenden (natürlich laufen danach keine ST-Programme unter MS-DOS, aber alle ASCII-Texte lassen sich z.B. prima weiterbearbeiten!).

Hans-Jürgen Merkel

```
' """"""""""""""""""""""""""""""""""""""""""""""""""""""""" '
'                Konvertierprogramm                    '
'                                             '
' um ST-formatierte Disketten für MS-DOS lesbar zu machen '
'                                             '
' Idee: Johannes Heyer (ST-Computer Nr. 2/90)          '
' Umsetzung in GFA-BASIC (V. 3.06): Hans-Juergen Merkel   '
' """"""""""""""""""""""""""""""""""""""""""""""""""""""""" '
GOSUB konvertieren
END
'
PROCEDURE konvertieren
  CLS
  inv_on$=CHR$(27)+"p"
  inv_off$=CHR$(27)+"q"
  meldung$=STRING$(4," ")+"Bootsektor ist
MS-DOS-lesefähig!"+STRING$(4,"
")
  puffer$=SPACE$(512)
  puffer_adresse%=V:puffer$
  t!=FALSE
  REPEAT
    PRINT AT(19,9);"Von welchem Laufwerk soll der Bootsektor"
    PRINT AT(19,10);"  MS-DOS-lesefähig gemacht werden ?"
    PRINT AT(27,12);"Laufwerk ";inv_on$;" A ";inv_off$;" oder
";inv_on$;"B ";inv_off$;" ?"
    PRINT AT(19,15);"Abbruch = >ESC<"
    PRINT AT(19,17);"Taste 'A' oder 'B' = Start"
    taste|=INP(2)
    IF taste|<>27
      IF taste|=65
        drive|=0
      ELSE
        drive|=1
      ENDIF
      r%=BIOS(4,2,L:puffer_adresse%,1,0,drive|)
      old_word$=HEX$(LPEEK(puffer_adresse%))
      new_word$="E9"+MID$(old_word$,3)
      IF r%
        ALERT 3,"Lesefehler !|",1,"Abbruch",dummy|
      ELSE
        {puffer_adresse%}=VAL("&H"+new_word$)
        r%=BIOS(4,3,L:puffer_adresse%,1,0,drive|)
        IF r%
          ALERT 3,"Schreibfehler !|",1,"Abbruch",dummy|
        ENDIF
        t!=TRUE
        FOR i|=1 TO 5
          PRINT AT(19,20);inv_on$;STRING$(40," ");inv_off$
          PAUSE 15
          PRINT AT(19,20);inv_on$;meldung$;inv_off$
          PAUSE 15
        NEXT i|
        PAUSE 50
      ENDIF
    ENDIF
  UNTIL t!=TRUE
RETURN
```

GFA-BASIC 3.5

Teil II

Im Artikel 'GFA-BASIC 3.5' der Club-Zeitschrift Nr. 2 '90 wurde auch über nicht vorhandene völlige Fehlerfreiheit gesprochen. Dieser erste Artikel ist ein praktisches Beispiel hierfür, denn die 'Textmaschine' bei GFA war nicht fähig, die Zeichen '<' und '>' aus meinem Text automatisch zu übernehmen.

Natürlich ist eine quadratische Matrix dann regulär, wenn die Determinante 'ungleich' Null ist; im Text wurde '<>' geschlabbert. Sorry!

Damit sind wir gleich beim Thema 'DETERMINANTEN'. Spezialisten möchten auch diesmal bitte nachsichtig reagieren. Im Bronstein, 'Taschenbuch der Mathematik', steht zur Determinante:

"Jeder n-reihigen quadratischen Matrix mit reellen bzw. komplexen Elementen läßt sich auf eindeutige Weise eine reelle bzw. komplexe Zahl zuordnen, die man als Determinante der Matrix bezeichnet."

Wie berechnet man nun die Determinante? Bei einer zweireihigen Matrix ist es mit 'Produkt der Hauptdiagonalen minus Produkt der Nebendiagonalen' idiotisch einfach. Die Hauptdiagonale geht von links oben nach rechts unten, die Nebendiagonale von rechts oben nach links unten.

Beispiel:

```
1 2 | V  1 mal 4  minus 2 mal 3 = -2
3 4 | Λ  a(1,1)*a(2,2)-a(1,2)*a(2,1)
```

Auch eine dreireihige Matrix kann nach der sogenannten Sarrusschen Regel noch relativ einfach errechnet werden. Zur optischen Verdeutlichung schreibt man die Spalten 1 und 2 rechts neben die Matrix. Dann werden die Summen der Produkte der Hautdiagonalelemente und der Elemente parallel zur Hauptdiagonalen gebildet und davon die Summen der Produkte der Nebendiagonalen ... subtrahiert. Man nennt dieses Verfahren auch 'Lattenzaun-Regel, weil die Diagonalen wie ein Lattenzaun über der Matrix liegen.

Am besten verdeutlicht das Listing 'determin.lst' - siehe unten - das hier Beschriebene, und durch das Experiment wird bekanntlich am besten gelernt.

Noch einmal zum obigen Beispiel: Das Vertauschen der Spalten verändert nur das Vorzeichen, nicht den Wert der Determinante selbst.

Die Determinante von quadratischen Matrizen mit mehr als drei Reihen könnten prinzipiell nach dem Schema ermittelt werden. Das wird aber dann sehr aufwendig. Hier geht man in der Regel nach dem Laplace'schen Entwicklungssatz vor. Er besagt, daß man eine n-reihige Determinante auf (n-1) reihige Determinanten zurückführt, bis man 3- bzw. 2reihige Determinanten erhält. Bei diesem Verfahren wird aus einer frei gewählten Spalte oder Zeile heraus entwickelt. Es ist dabei gleichgültig, welche Spalte oder Zeile gewählt wird. Zwar sind die Zwischenergebnisse naturgemäß unterschiedlich, das Endergebnis - die Determinante - ist immer gleich.

Da es sich hier um eine Einführung für Ein- und Aufsteiger handelt und die Erläuterungen einen weiteren, eigenen Artikel erfordern würden, verweise ich auf Lehrbücher und sonstiges Lehrmaterial. Wer z.B. einen Mathematiker oder Informatiker in seinem Bekanntenkreis hat, der kann sich das ja mal erklären lassen, viel Spaß. Ich bedanke mich hier recht herzlich bei Michael Kauke für seine Geduld.

```
.he   determin.gfa   khs 5k40 für gfa club-nachrichten mai/juni 90  \d
.n9
OPTION BASE 1
MAT BASE 1
weiter:
ERASE a(),b(),c()
txt$=" welche Matrix-Gröe? "
ALERT 0,txt$,0," 2x2 | 3x3 ",dum|
IF dum|=1
 x=2
ELSE
 x=3
ENDIF
DIM a(x,x),b(x,x),c(x,x)
txt$="variable oder feste Vorgabe ? "
ALERT 0,txt$,1,"variabel|DATA's",dum|
IF dum|=2                              ! fest aus datas
 IF x=2
  RESTORE zwei
 ELSE
  RESTORE drei
 ENDIF
 MAT READ a()
ELSE
 FOR i=1 TO x
  FOR j=1 TO x
   a(i,j)=RAND(x+6)                    ! einlesen mit
  NEXT j                               ! zufallszahl
 NEXT i
ENDIF
PRINT " MATRIX a(";x;",";x;")"
MAT PRINT a(),4,0
MAT DET d=a()
IF d=0
 PRINT STRING$(45,"*")
 PRINT " DET a.d. Koeffiz. = ";d;" , also keine INVERSE";CHR$(7)
 PRINT
 ALERT 0,"",1,"weiter|ende",dum
 IF dum=1
  CLS
  GOTO weiter
 ENDIF
 EDIT
```

```
ENDIF
IF x=3
  @lattenzaun
ENDIF
PRINT
PRINT " INVERSE von a() mit MAT INV b()=a()"
MAT INV b()=a()
MAT PRINT b(),8,4
PRINT
PRINT " prüfen auf EINHEITSMATRIX"
PRINT " durch   a() * INV von a()"
MAT MUL c()=a()*b()
MAT PRINT c(),5,2
PRINT AT(2,16);STRING$(62,"_")
IF x=2
  @dethandzwei
ENDIF
IF x=3
  @dethanddrei
ENDIF
ALERT 0,"",1,"neu|ende",dum
IF dum=1
  CLS
  GOTO weiter
ENDIF
EDIT
' ----------
PROCEDURE dethandzwei
  dp=a(1,1)*a(2,2)
  dm=a(1,2)*a(2,1)
  PRINT AT(2,18);"a(1,1)*a(2,2) - a(1,2)*a(2,1)"
  PRINT AT(5,19);a(1,1);"  *  ";a(2,2);"
                  - ";a(1,2);"  *  ";a(2,1)
  PRINT AT(1,21);" Determinante a(";x;",";x;") GfA 3.5 = ",d
  PRINT AT(1,22);" Determinante a(";x;",";x;") manuell
                                              =",dp-dm
RETURN
'
PROCEDURE dethanddrei
  dp=a(1,1)*a(2,2)*a(3,3)+a(1,2)*a(2,3)*a(3,1)+a(1,3)
                                          *a(2,1)*a(3,2)
  dm=a(1,3)*a(2,2)*a(3,1)+a(1,1)*a(2,3)*a(3,2)+a(1,2)
                                          *a(2,1)*a(3,3)
  PRINT AT(2,18);"a(1,1)*a(2,2)*a(3,3)+a(1,2)*a(2,3)
                   *a(3,1)+a(1,3)*a(2,1)*a(3,2)"
  PRINT AT(2,19);" = + ",,dp
  PRINT AT(2,20);"a(1,3)*a(2,2)*a(3,1)+a(1,1)*a(2,3)*a(3,2)
                                       +a(1,2)*a(2,1)*a(3,3)"
  PRINT AT(2,21);" = - ",,dm
  PRINT AT(2,23);"Determin. a(";x;",";x;") GfA 3.5 = ",d
  PRINT AT(2,24);"Determin. a(";x;",";x;") manuell = ",dp-dm
RETURN
'
PROCEDURE lattenzaun    ! noch print using**
  us1$="|"+"-##"
  us2$="-##"
  PRINT AT(16,1);"| Hilfsspalten 1 u.2 für 'Lattenzaun'"
  PRINT AT(16,2);USING us1$,a(1,1)
  PRINT AT(21,2);USING us2$,a(1,2)
  PRINT AT(16,3);USING us1$,a(2,1)
  PRINT AT(21,3);USING us2$,a(2,2)
  PRINT AT(16,4);USING us1$,a(3,1)
  PRINT AT(21,4);USING us2$,a(3,2)
RETURN
'
zwei:
' DATA 0,1,0,2
DATA 1,2,3,4
drei:
' DATA 1,2,3,4,5,6,7,8,9
DATA -2,-5,1,1,3,-6,3,21,-15
' DATA 5,4,0,0,5,5,2,0,1
```

Zur Abwechslung einmal ein Beispiel zu den neuen Befehlen aus der Kombinatorik, hier COMBIN.

Wer genau wissen will, wie gering die Chancen beim Otto vom Lotto sind, kann das mit Hilfe von Fakultäten errechnen. Zum Beispiel für 6 richtige Zahlen aus 49 mit:

Fakultät von 49/Fakultät von (49-6)*Fakultät von 6.

Das nachfolgende Listing 'Lotto.lst' zeigt den Programmablauf ohne spezielle Befehle. Auch wenn die Funktion Fakultät vorhanden ist, bleibt noch ein gewisser Eingabeaufwand.

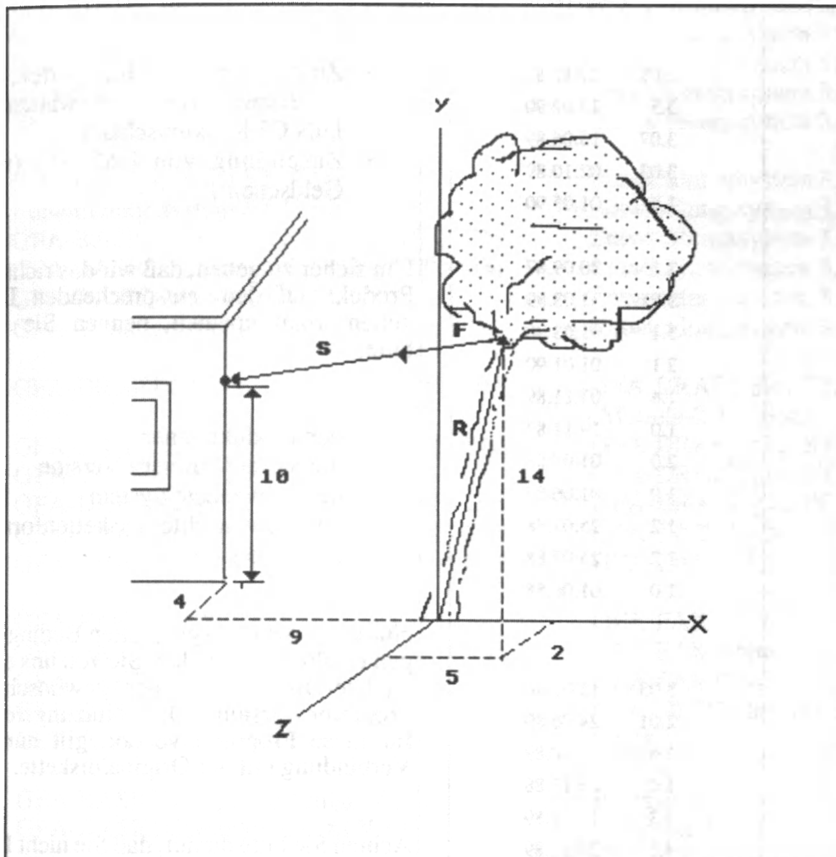
Mit der Funktion COMBIN(n,k) sind in einer Zeile nur noch n und k (= n über k) einzugeben. So bleibt Zeit zum Denken. Wer den Spaß und den Kitzel des Glücksspieles haben will, für den reicht eine Zeile auf dem Lottoschein wie im Programm, den Rest kann er in andere Dinge - z.B. gute Software - investieren.

Natürlich ist die Frage nach den Lotto-Chancen nicht die einzige und vor allem nicht die wichtigste aus dem Gebiet der Kombinatorik.

```
.he lotto.lst  khs 5k40 für gfa club-nachrichten mai/juni 90   \d
.n9
PRINT " aus der Kombinatorik"
PRINT " z.B. Lotto 6 aus 49 wird berechnet durch :"
PRINT !------------------------------------------------------------
PRINT " GFA 3.5 in einer Zeile durch  ==>  COMBIN(49,6)
                                              =",COMBIN(49,6)
PRINT !------------------------------------------------------------
PRINT " manuell durch folgende Formel / und Programm :"
n1=49
n2=6
fakultaet(n1)
erg=x
fakultaet(n1-n2)
erg=erg/x
fakultaet(n2)
erg=erg/x
PRINT " Fakultät von 49/Fakultät von 49-6*Fakultät von 6 = ",erg
~INP(2)
EDIT
'
PROCEDURE fakultaet(n)
  x=1
  FOR i=1 TO n
    x=x*i
  NEXT i
RETURN
' ------------------------------------------------------------
```

Nun für heute zum dritten und letzten Abschnitt des Artikels. Das Beispiel habe ich aus einem alten HP-Handbuch. Dort stand eine Funktion MAT (Ergebn.Vcrktor)=CROSS (Op-Vekt.1,Op-Vekt.2). Damit wurde das Kreuz- oder Vektor-Produkt zweier Vektoren mit jeweils drei Elementen berechnet.

Beim Testen von GFA 3.5 reizte mich, wie das Problem ohne diese Funktion zu lösen ist. Das Ergebnis finden Sie im dritten und letzten heutigen Listing kreuzprodukt.lst am Ende des Artikels.




```
.he xprodukt.lst    khs 5k40 für gfa-club ausg. mai/juni 90  \d
.n9
DIM r(3,1),f(3,1),s(3,1),m(3,1),l(1,1)
DATA 5,14,2,-14,-4,-6
MAT READ r()
MAT READ s()
PRINT " R(3,1) ist Richtungsvektor des Haltepunkts a rel. z. Fu d. HP a"
MAT PRINT r(),4,0
PRINT STRING$(62,".")
PRINT " S(3,1) ist der Vektor vom Haltepunkt a zum Haltepunkt b"
MAT PRINT s(),4,0
PRINT STRING$(62,".")
PRINT " L(1,1) ist Ergebnis der Berechnung der Länge von S + 960Kg/L"
l(1,1)=960/SQR(s(1,1)^2+s(2,1)^2+s(3,1)^2)
MAT PRINT l(),14,9
PRINT STRING$(62,".")
PRINT " Berechnung der Komponenten von F(3,1) ==> MAT MUL f()=s()*l()"
MAT MUL f()=s()*l()
MAT PRINT f(),10,4
PRINT STRING$(62,".")
PRINT " Kraftmoment ermittelt durch   Kreuzprodukt M = R X F"
PRINT " (HP MAT Ergebnisvektor = CROSS (Operandenvekt.1,OPV2))"
m(1,1)=r(2,1)*f(3,1)-r(3,1)*f(2,1)
m(2,1)=r(3,1)*f(1,1)-r(1,1)*f(3,1)
m(3,1)=r(1,1)*f(2,1)-r(2,1)*f(1,1)
MAT PRINT m(),10,2
PRINT AT(12,21);" x-Komponente     ==> Mx = Ry*Fz - Rz*Fy"
PRINT AT(12,22);" y-Komponente     ==> My = Rz*Fx - Rx*Fz"
PRINT AT(12,23);" z-Komponente     ==> Mz = Rx*Fy - Ry*Fx"
~INP(2)
EDIT
```

Zum abgebildeten Beispiel:

Ein schrägstehender Baum wird durch ein Stahlseil mit der Ecke des Hauses (siehe Bild) verbunden.

Die Fragestellung lautet:
Wie stark ist das Kraftmoment, das auf das Verbindungskabel über dem Fuß des Baumes ausgeübt wird, wenn die Spannung im Draht 960 kg beträgt?

Lösung:
Das Kraftmoment wird beschrieben durch das Kreuzprodukt

M = R x F

R ist der Richtungsvektor des Haltepunktes (am Baum) relativ zum Fuß, F sind die 960 kg Kraft, die auf das Verbindungskabel ausgeübt werden.

R, S und F werden durch ihre Komponenten in x-, y- und z-Richtung beschrieben. S ist dabei der Vektor vom Haltepunkt des Baumes zum Haltepunkt am Haus. Die Komponenten von R:

Rx=5, Ry=14, Rz=2

Die Komponenten und Längen von F u. S sind proportional:

Fx/Sx = Fy/Sy = Fz/Sz = F/S

Daher kann jede Komponente von F als Produkt der entsprechenden Komponente von S mit dem Quotienten der Längen von F und S berechnet werden.

Für die Komponenten von S erhält man:

Sx= -9 -5=-14 Sy= 10-14=-4 Sz=-4-2=-6

(siehe Bild und Datas im Listing)

Zugegeben, nicht jeder hat einen schrägstehenden Baum, der darüber hinaus noch mit der eigenen Hütte verbunden ist. Aber es gibt im täglichen Leben so viele sich gegenseitig beeinflussende Kräfte, die von uns ebenfalls auszuhalten sind. Für heute reicht's mir, alles Gute!

Karl-Heinz Schulze

Neueste Versionen

Name	Version	Datum
ST-Produkte		
GFA-BASIC		
Entwicklungssystem 2.0 (Interpreter und Compiler)	2.02	27.12.87
Entwicklungssystem 3.5 (Interpreter und Compiler)	3.5	14.02.90
Interpreter 3.0	3.07	15.06.89
Compiler 3.0	3.03	02.10.89
GFA-ASSEMBLER	1.5	01.05.90
GFA-DRAFT (an Mega ST angepaßt)	2.1	30.09.87
GFA-DRAFT-plus ST	3.01	31.03.89
GFA-DRAFT-plus ST	3.1	01.03.90
GFA-RAYTRACE	2.1	01.01.90
ChemGraf	1.4	03.11.89
GFA-ARTIST	1.0	19.11.87
STARTER	2.0	01.07.88
Floppyspeeder	1.0	01.06.88
Farbkonverter	1.2	25.07.88
Monochromkonverter	1.2	25.07.88
Multi-Accessory	1.0	01.06.88
PC-Produkte		
GFA-DRAFT-plus	3.03	15.01.90
GFA-FAKT 2.0	2.01	24.06.89
GFA-FAKT-plus	1.6	02.10.89
GFA-DESK	1.6	05.11.88
GFA-DESK-plus	1.3	17.07.89
GFA-CAD	4.5	27.11.89
AMIGA-Produkte		
GFA-BASIC Interpreter 3.0	3.05	04.09.89
GFA-BASIC Compiler 3.0	3.01	01.12.89
GFA-BASIC Interpreter 3.5	3.5	01.02.90
GFA-BASIC Compiler 3.5	3.5	01.02.90
GFA-ASSEMBLER	1.5	01.03.90

Service

Sie müssen registrierter Kunde für das "upzudatende" Produkt sein. Das Update erfolgt nach:

- Zusendung eines frankierten, an Sie selbst adressierten, wattierten DIN C5 Rückumschlages
- Zusendung von DM 10.- (nur Geldschein)

Um sicher zu gehen, daß wir das richtige Produkt auf dem entsprechenden Diskettenformat updaten, nennen Sie uns bitte:

- den Produktnamen
- die Seriennummer / Version
- das verwendete System
- das gewünschte Diskettenformat (5 1/4" oder 3 1/2")

Nur wenn die oben genannten Bedingungen erfüllt sind, erhalten Sie von uns eine Update-Diskette mit der gewünschten Programmversion. Das Nutzungsrecht für diese Programmversion gilt nur in Verbindung mit der Originaldiskette.

Achten Sie bitte darauf, daß Sie nicht Ihre Originaldiskette einsenden!

Supportzeiten bei der GFA

GFA-BASIC Atari ST	Dienstag:	15.00 bis 17.00 Uhr
GFA-BASIC Amiga	Dienstag:	15.00 bis 17.00 Uhr
GFA-DRAFT-plus PC/ST	Montag:	15.00 bis 17.00 Uhr
und GFA-CAD	Donnerstag:	15.00 bis 17.00 Uhr
GFA-ASSEMBLER, GFA-BASIC ST	Mo, Mi, Fr:	9.00 bis 13.00 Uhr
	Di und Do:	13.30 bis 17.00 Uhr
GFA-FAKT-plus	Montag:	10.00 bis 11.30 Uhr
GFA-DESK-plus, GFA-FAKT 2.0	Freitag:	10.00 bis 11.30 Uhr

Upgrades

Programm	Preis	Upgrade	Preis	Bemerkung
GFA-BASIC Interpreter ST V 2.0	169,- DM	Entwicklungssystem 3.0	70,- DM	1 + 6
GFA-BASIC Interpreter ST V 2.0	99,- DM	Entwicklungssystem 3.0	140,- DM	1 + 6
GFA-BASIC Interpreter V 1.0		Entwicklungssystem 3.0	90,- DM	1
GFA-Einsteiger-Set		Entwicklungssystem 3.0	140,- DM	4
Entwicklungssystem ST V 2.0		Entwicklungssystem 3.0	160,- DM	1
Entwicklungssystem ST V 2.0		Entwicklungssystem 3.5	230,- DM	1
Entwicklungssystem ST V 3.0		Entwicklungssystem 3.5	70,- DM	2
GFA-BASIC Interpreter ST V 2.0	169,- DM	Entwicklungssystem 3.5	140,- DM	1 + 6
GFA-BASIC Interpreter ST V 2.0	99,- DM	Entwicklungssystem 3.5	210,- DM	1 + 6
GFA-BASIC Interpreter V 1.0		Entwicklungssystem 3.5	160,- DM	1
GFA-Einsteiger-Set		Entwicklungssystem 3.5	210,- DM	4
GFA-DRAFT ST	298,- DM	GFA-DRAFT plus V3.0 (Modula-2-Version)	120,- DM	3
GFA-DRAFT ST	198,- DM	GFA-DRAFT-plus ST V 2.0	220,- DM	3
GFA-DRAFT-plus ST		GFA-DRAFT-plus ST V 3.1	50,- DM	2
GFA-DRAFT-plus PC V 3.01	279,- DM	GFA-DRAFT-plus PC (Er. V.)	99,- DM	3
GFA-DRAFT-plus PC V 3.01	279,- DM	GFA-CAD V 4.5	499,- DM	1
GFA-DRAFT-plus PC Erw. Ver.	398,- DM	GFA-CAD V 4.5	400,- DM	1
GFA-FAKT V 1.X		GFA-FAKT V 2.0	25,- DM	3
GFA-FAKT V 1.X		GFA-FAKT-plus	425,- DM	3
GFA-FAKT V 2.0		GFA-FAKT-plus	400,- DM	3
GFA-DESK V 1.5		GFA-DESK plus V 1.2	70,- DM	2
GEM (GFA-DRAFT-plus PC) V 2.2		GEM 3.1	149,- DM	5
GFA-BASIC Interpreter Amiga V 3.0		Interpreter 3.5	30,- DM	2
GFA-BASIC Compiler Amiga V 3.0		Compiler 3.5	40,- DM	2

Bemerkung: (Folgendes bitte einschicken)

1: komplettes Paket (Handbuch, Ringordner u. Orig.-Diskette)
2: Originaldiskette
3: Handbuchseiten + Originaldiskette
4: nichts
5: nur Masterdiskette
6: Kopie des Kaufbelegs

Messetermine bis Juli '90

2.- 9.5.1990	Hannover	**HANNOVER MESSE Industrie**
29.-31.5.1990	Sindelfingen	**IDENT / VISION** Intern. Fachmesse für industrielle Bildverarbeitung und künstliche Intelligenz
29.5.-1.6.1990	Stuttgart	**CAT** Computerunterstützte Technologien - Intern. Fachausstellung und Anwendungskongreß

Die Public Domain Box

Wie versprochen, hier ist wieder die Vorstellung der neuen PDs. Im Laufe des letzten Monats sind bei mir wieder ein paar Disketten für unsere PD-Box eingetroffen, die ich heute vorstellen möchte.

Anwendungen

Für all diejenigen, die sich über unsere Desktop-Druckroutine ärgern, haben wir nun einen kleinen Leckerbissen anzubieten. Dank Uwe Ritterstädt hat die fast unnachahmliche Verstümmelung unserer Texte durch die Desktop-Printroutine nunmehr endlich ein Ende. Neben der angenehmen Druckgeschwindigkeit erkennt sein Programm die Schriftattribute von 1st Word plus sowie die Sonderzeichen und Umlaute. Auch haben Sie jetzt endlich die Möglichkeit, NQL oder Draft sowie die Fontarten Elite, Draft oder Italic über den Computer zu wählen, wobei auch der Grundschrifttyp nicht vergessen wurde. Da der Autor uns neben dem Programm auch den GFA-BASIC 3.x Sourcetext geschickt hat, kann das Programm auch leicht um weitere Textarten wie Signum2 oder Wordperfect erweitert werden. Print ist zusammen mit einem Balkengrafikprogramm und allen Quelltexten auf der PD #145 erhältlich und benötigt den SM 124.

Fun

Stonewatch mit Willy kennt wohl jeder Atarianer. Nun haben wir aber eine wesentlich schnellere Version dieses Oldies auf Lager. Das Spiel, das auf dem Monochrome-Monitor läuft, beinhaltet auch einen guten Editor, mit dem eigene Levels erstellt oder alte verändert werden können. Da die Levels alle etwas knifflig sind, werden wohl die meisten User erst mal frustriert zum Editor greifen und einige der bösen Fallen entfernen (ich gestehe, auch ich habe geschummelt!). Unser Spiel des Monats hat die Nummer #146 und wird nur auf eigene Gefahr abgegeben!

Utilities

Natürlich waren auch unsere Freunde aus Holland wieder aktiv. "11 nuttige Programmas" (Orginaltext Holland, 11 nützliche Programme) wie Madonna, Catalog, Desked4, Diskview, Matrix, Postbank, Prim oder WDWFunkt machen ab heute den Umgang mit den Disketten und dem Desktop easy. Holland sei für unsere Diskette #143 gedankt. Mal sehen, was beim nächsten Mal von Euch kommt.

Grafik-Demo

Auch der Club aus Belgien (schöne Grüße) war diesen Monat nicht untätig. Von ihnen stammt das Tristar-Demo, das von 90! (in Worten NEUNZIG) Zeichenfunktionen nur drei gesperrt hat. Tristar gehört meiner Meinung nach zu den besten Zeichenprogrammen und lohnt sich auch als Demo. Im Vertrauen, Tristar-Demo hat die Nummer #149. Aber nicht weitersagen!

Grafik/RSC

Da wir schon mal bei den Zeichenprogrammen sind, LOW ist unser neues PD-Zeichenprogramm und bietet viele tolle Funktionen und Möglichkeiten zum Erstellen von guten Grafiken. Da man von Grafiken alleine nicht satt wird, haben wir auf dieser Diskette auch ein Programm zum Herstellen von RSC-kompatiblen Icons beigelegt. TOOLBOX beinhaltet alles (und vieles mehr), um leicht Icons und Defblocks zu erstellen. Toolbox glänzt, wie auch LOW, mit einer gut durchdachten Benutzeroberfläche und schöner Bedienbarkeit. Beide Programme warten auf Diskette #139 auf Euch!

Geld/Musik

Ein sehr leidiges Thema bei uns Computerfreaks ist das Geld. Da hat der Laserprinter ein Riesenloch in die Kasse gebrannt, wo doch gerade eben die Megafile 44 vom letzten Monat verkraftet war. So etwas wäre mit unserer Diskette #141 nicht passiert. Zwar kann die Diskette keine Tausender drucken (leider), aber mit dem Haushaltsprogramm auf der Diskette haben wir die nötige Übersicht über unser Geld. Hiermit können wir die Einnahmen den Ausgaben (oh Schreck!) gegenüberstellen und sehen, wo wir uns ausnahmsweise mal bremsen müssen. Falls Sie dann trotzdem zuviel Geld ausgegeben haben, können Sie mit SYNTHY einen Trauermarsch komponieren und abspielen. Für ungeübte Komponisten haben wir ein paar kleine Songs beigelegt, die auch nach einem Finanzkoller zur gehobenen Stimmung beitragen könnten.

Textsysteme

Für Textakrobaten habe ich auch mal wieder was zu bieten! Dank eines TOS-Texteditors sowie eines komplett in GEM eingebundenen Texteditors, beide sowohl für Farbe als auch für Schwarz/Weiß geeignet, können sich unsere Mitglieder nicht mehr mit 'mangelnden' Textsystemen für das Fehlen von Artikeln entschuldigen. Auf geht's Leute, ran an's Keybord! Mit diesen Programmen könnt Ihr endlich Artikel am Fließband produzieren, denn in der nächsten Ausgabe ist noch Platz! Unsere Textsystemdisk hat die Nummer #150!

Assembler-Paket

Ein kleiner Leckerbissen für Assembler-Neulinge! Der Devpac-Assembler in seiner 1. Version als PD. Ist doch schon was, oder? Zwar kann man diesen Assembler nicht mit dem GFA-ASSEMBLER vergleichen, denn hier hinkt sowohl die Geschwindigkeit als auch die Bedienungsfreundlichkeit stark hinterher, aber für ein oder zwei Versuche genügt's allemal. Wie Ihr den GFA-ASSEMBLER bestellen könnt, wißt Ihr ja, oder? Unser Assemblerpaket wartet auf der Diskette #152 auf Euch!

Assemblerkurs

Wie versprochen, ist hier die Diskette zu unserem Assemblerkurs mit den Listings und Erklärungen zu dem Kurs. Nebenbei sind die Betriebssystemfunktionen auch schon vordefiniert, so daß Ihr gleich loslegen könnt, wenn Ihr die Diskette #153 bestellt! (Der GFA-ASSEMBLER wäre hierzu auch empfehlenwert, denn die Listings sind mit dem GFA-ASSEMBLER geschrieben worden!)

So, das wäre es für heute mal wieder. Ich hoffe, es war wieder etwas für Euch dabei. Und falls Ihr mal einen Gesamtüberblick über unsere PD-Box haben wollt, die neue PD-Liste ist noch warm und wartet auf Euch. Toll wäre es natürlich, wenn bei dieser Anforderung gleich eine neue PD beiliegen würde. Mein Dank sei Euch schon heute gewiß.

(rk)

Flohmarkt-Anzeigen

Biete

Atari 1040 STF, TOS 1.0 in ROM, incl. Maus, Omikron-Basic, Handbücher, technisch und optisch einwandfrei, 5 Jahre alt, OHNE Monitor: nur 500,- DM.
Atari SH 205, 20 MB Festplatte, technisch und optisch einwandfrei, wurde vor dem Ausschalten immer geparkt, 2 Jahre alt: nur 700,- DM.
Dr. T. Rubach, Kurfürstenstr. 36, 8034 Germering, Tel.: 089 / 846030.

Biete an: Kombinierte Entwickler-Shell für alle GFA-BASIC-Systeme und GFA-ASSEMBLER für nur 20,- DM. Viele Utilities, mehr Info gegen Rückumschlag. D. Redanz, Kirchstr. 91, 4353 Oer-Erkenschwick.

Verkaufe für Atari ST: Minigolf 20,- DM, Great Courts 45,- DM, Protos 50,- DM, Diskettenverwaltung Diskdat ST 20,- DM, Schallplattenverwaltung (LP, CD, MC) 1St-Sound 20,- DM, alles Originale! Tel.: 08341 / 800223 oder 08349 / 625.

Amiga Originalspiele Powerdrome und Katakis je 30,- DM, Deluxe Music Construction Set mit Midi und Demodisk 90,- DM, Textomat und Datamat zusammen 79,- DM. Mehrere Bücher zum Amiga, je halber Originalpreis. Amigas Harddisk 20 MB 85,- DM. Jede Menge PD-Disketten, z.B. Fred-Fish, Kickstart, RPD, Franz, Cactus, Taifun usw., über 1500 Stück, je Disk nur 2,40 DM. Einige Leerdisketten: 10 Stück für 14,- DM. Helmut Dalege, Basilikastr. 15, 4178 Kevelaer 1, Tel.: 02832 / 7425 nach 17 Uhr.

Verkaufe: GFA-BASIC ST 3.0 EWS (NP 198,- DM) für 140,- DM; Atari 520 STM mit Floppy SF 354 VB 490,- DM; Atari SM 124 VB 280,- DM; GFA-BASIC-Buch 3.0 (incl. Disk), Heim-Verlag VB 25,- DM. Hans Hundhammer, 8215 Marquartstein, Tel.: 08641 / 621777.

TRIUMPH-ADLER alphatronic PC 150,- DM, ECD-Bus Adapter (aus c't 10/87) 60,- DM. Tel.: 05604 / 5374 ab 18 Uhr.

Verkaufe: SKYBLASER 20,- DM, ST-WARS 20,- DM, JUMP JET 20,- DM, AFTER BURNER 25,- DM, ELITE 30,- DM, FALCON F16 40,- DM, ZAK MAC KRACKEN 35,- DM. Alles Originale mit Anleitung usw. Rufe an: 06424 / 4504, K.-H. Wallon, Leidenhofenerstr. 25, 3557 Ebsdorfergr. 6

Suche

Suche INDIANDER JONES usw. als Abenteuerspiel und MANIAC MANSON, beides als Original mit allem Drum und Dran. Ruft an: 06424 / 4504.

Für GFA-Club-Mitglieder kostenlos: Ihre persönliche Kleinanzeige für den Flohmarkt
Bitte veröffentlichen Sie in der nächstmöglichen Ausgabe der GFA-CLUB-Nachrichten unten folgenden Text (auch als ASCII-File auf Diskette möglich)! Für jegliche Anzeigen übernehmen wir keine speziellen Formatierungsarbeiten.

Biete an ☐ Suche: ☐ Kontakte: ☐ Verschiedenes: ☐

Maximal 15 Zeilen, je Zeile 30 Zeichen (inkl. Zwischenraum) bei normaler Schriftgröße. Bei Fettdruck und grafischen Zeichen sind Abweichungen möglich.

Name

Strasse

PLZ/Ort

Datum/Unterschrift

Nicht-Club-Mitglieder: 1.- DM / Zeile 5.- DM / Zeile / gewerbliche Anzeigen
Club-Mitglieder: 3.- DM / Zeile / gewerbliche Anzeigen

Club-Mitglied: ☐
Nicht-Club-Mitglied: ☐
gewerbliche Anzeige: ☐

Der Anzeigenpreis in Höhe von ______ DM liegt bei als V-Scheck: ☐ bar: ☐

Take off, ATARI!

Hier ist nun der erste Teil unseres Assemblerkurses für Einsteiger. In dieser Folge wollen wir uns mit den elementaren Seiten der Maschinensprache auseinandersetzen und ein GFA-BASIC-Programm von Hand übersetzen. Als Handwerkszeug haben wir uns für den GFA-ASSEMBLER entschieden, da dieser neben der einfachen Bedienbarkeit auch noch mit dem GFA-BASIC 3.x harmoniert. Obendrein kann mit ihm das fertige Programm sofort getestet werden, ohne daß der Assembler erst verlassen werden muß.

Was ist eigentlich Assembler? Assembler oder Maschinensprache ist die unterste Ebene des Programmierens und stellt eine Sprache dar, die der Computer direkt ohne Compiler oder Interpreter (Dolmetscher) versteht und ausführen kann. Im Gegensatz zu Hochsprachen besitzt die Maschinensprache nur eine Handvoll Befehle, so daß auch einfache Dinge wie "PRINT" durch eigene Unterprogramme oder Systemroutinen erstellt werden müssen. Aber halt, ich greife ja mal wieder voraus, denn erst sollten wir uns mal mit dem für uns wichtigsten Teil des Prozessors, den Registern (siehe Abbildung), beschäftigen.

Da ohne die Register bei unserem ATARI gar nichts läuft, wollen wir hier mal etwas ausführlicher werden. Als erstes wäre hier der Programmzähler (PC, engl. program counter) zu nennen. Er beinhaltet immer die Adresse des nächsten Befehls und umfaßt 32 Bytes, wobei jedoch nur 24 Bytes zum Tragen kommen. Ein paar Befehle sprechen den PC direkt an, wobei wir als Programmierer jedoch nur selten mit Adressberechnungen über den PC zu tun haben werden.

Ein Register, ohne dessen Manipulation wir kein vernünftiges Programm zum Laufen bringen würden, ist hingegen das Statusregister (SR). Dieses in den System- und Userteil unterteilte Register beinhaltet wichtige Zustände sowohl des Systems als auch unseres Programmes. So kann hier abgefragt werden, ob eine Operation mit einem Nullergebnis endet oder gar negativ ist und in welchem Modus sich das Programm gerade befindet.

Ein äußerst wichtiges Register ist der Strackpointer (A7 oder SP). Da der MC68000 zwei verschiedene Arbeitsmodi kennt, existieren von diesem Register auch zwei Ausgaben, wobei wir uns in diesem frühen Stadium nur mit dem sogenannten Userstrackpointer beschäftigen wollen. Der Strackpointer, dt. Stapelzeiger, stellt einen reservierten Speicherbereich dar, in dem Parameter und Unterprogrammadressen sowie Rücksprungadressen für spätere Aktionen zurückgelegt werden. Dieser Strack ist im Gegensatz zu so manchem 8-Bit Computer ein LIFO-Strack (LIFO= Last In, First Out), bei dem die Daten wie auf dem Schreibtisch einfach oben drauf gepackt werden. Will man jetzt an Daten darunter kommen, muß von oben weg gearbeitet werden. Noch etwas erinnert beim Strack an den Schreibtisch: Wie auf unseren Schreibtischen müssen wir auch hier selbst für Ordnung sorgen; wenn wir dies unterlassen oder einfach vergessen, droht uns der ATARI mit einem saftigen Absturz (ähnlich wie mein Schreibtisch, wenn ich nicht gleich aufräume)!

Neben diesen Registern besitzt der 68000'er noch 15 'freie' Register, die wir für Daten und Adressen verwenden können. Diese werden D0-D7 für Datenregister und A0-A6 für Adressregister genannt, doch dazu später mehr.

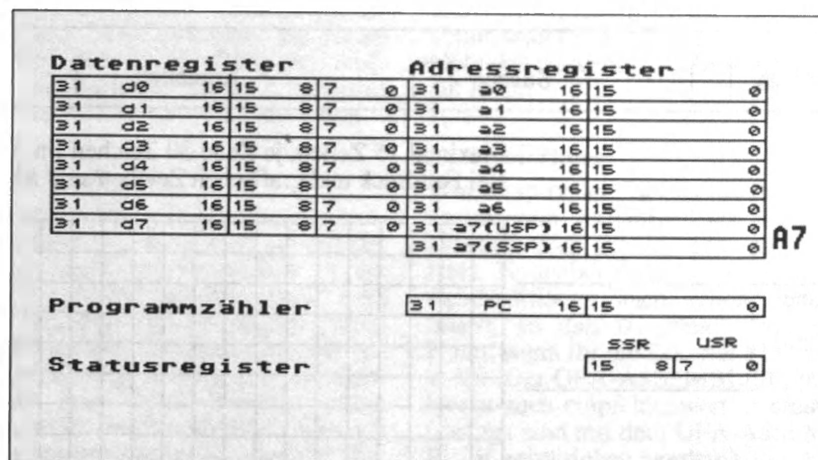
Manche werden jetzt wegen der Masse an Registern stöhnen, diese 'Menge' ermöglicht jedoch eine sehr hohe Geschwindigkeit bei Daten und Adressregistermanipulationen oder Speicherverschiebungen. Doch grau ist alle Theorie, fangen wir mal mit dem Einstieg an:

Wie schon erwähnt, wollen wir zum langsamen und gemütlichen Einstieg in die Welt des Prozessors ein GFA-BASIC 3.5 Programm von Hand in die Assembler-Sprache übersetzen. Zum einfacheren Verständnis sollten Sie sich das untenstehende Programm einmal anschauen und eventuell auch ausprobieren.

```
FOR V%=0 TO 399            ! Vertikale Auflösung
  FOR H%=0 TO 639          ! Horizontale Auflösung
    PLOT H%,V%             ! Setze Bildpunkt
  NEXT H%                  ! NEXT Horizontal
NEXT V%                    ! NEXT Vertikal
```

Diese Basic-Routine zum 'Schwärzen' des Bildschirms benötigt genau 3 min und 8 sec zur Ausführung und ist in der Praxis eigentlich nicht zu verwenden. Da sie jedoch einfach zu verstehen ist und einen guten Einstieg bietet, haben wir uns für sie entschieden. Nebenbei treten hier die Geschwindigkeitsunterschiede auch besonders zutage. Doch zurück zum Assembler.

Nach dem Start vom GFA-ASSEMBLER werden Sie nach dem Filenamen gefragt, der auf .IS enden sollte.



Der Einfachheit halber wollen wir unser Assemblerlisting hier mal BLACK.IS nennen und dann <RETURN> drücken. Falls Sie sich das Abtippen ersparen wollen, unsere PD-Box beinhaltet nun auch den Assemblerkurs mit allen Files und Erklärungen. Nach dem Start befinden wir uns im Editor des Assemblers und könnten rein theoretisch schon loslegen. Zuvor

jedoch etwas zum Aufbau einer typischen Befehlszeile im Assembler. Eine solche Zeile sieht so aus:

```
START:   move.w #2,-(SP)  ; Funktionsnummer
                            Physbase (XBIOS 2)
```

START stellt ein LABEL dar, das vom Assembler hinterher durch eine Adresse ersetzt wird. Labels müssen im GFA-ASSEMBLER immer mit einem Doppelpunkt enden und in der ersten Spalte stehen. Allgemein gilt auch, daß die Labels acht Stellen lang sein dürfen und fast alle Begriffe erlaubt sind.

Direkt hinter dem Label, in der zweiten Spalte, steht das Befehlswort mit der Verarbeitungsbreite, in unserem Fall 'move.w' also 'Bewege mit Wortlänge', gefolgt von der Datenquelle und dem Datenziel. Zum besseren Verständnis von Programmen können in der letzten Spalte auch noch Kommentare eingefügt werden, die jedoch durch ein Semikolon abgetrennt sein müssen. Doch jetzt wieder zum Programm.

Unser Basic-Programm setzt in der hohen Auflösung jeden Punkt des Bildschirms auf Schwarz. Um dies mit einem Assemblerprogramm zu erreichen, müssen wir zuerst die Adresse des Bildschirms feststellen und sichern. Da unser Betriebssystem (BS) recht gut aufgebaut ist und wir das Rad im Moment noch nicht neu erfinden wollen, fragen wir es nach dieser Bildschirm-Adresse. Um dem BS, das aus GEMDOS, BIOS, XBIOS, AES und VDI besteht, zu sagen, was wir denn wollen, haben sich die Entwickler auf einen Zahlencode geeinigt, bei dem jede Zahl eine Funktionsnummer des BS darstellt. Zu diesen Funktionsnummern möchten wir Ihnen das GFA-Buch TOS&GEM oder das ATARI ST Profibuch von Sybex empfehlen, da hier alle Funktionen aufgeführt sind.

In unserem Fall benötigen wir die Funktionsnummer 2 vom XBIOS (Extendet BIOS), denn diese übergibt uns die Adresse des Bildschirms. Auch die Übergabe unserer Parameter an das BS ist genormt, denn es muß unsere Eingabe ja auch verarbeiten können. Hier an dieser Stelle kommt dann der Strack und unser erster Befehl zum Einsatz:

```
move.w   #2,-(sp)  ; XBIOS Funktion 2
```

Wir bewegen über den Move-Befehl die Zahl 2 direkt auf den vorher verringerten Strack. Das .w hinter move zeigt uns die Verarbeitungsbreite des Befehls an, der hier mit Word-Länge (16 Bit) arbeitet. Genausogut kann move auch 8 Bit (.b) oder 32 Bit (.l) bewegen. In unserem Fall benötigen wir jedoch nur 16 Bit.

Das Doppelkreuz vor der Zahl 2 zeigt an, daß diese Zahl direkt übergeben wird. Fehlt dieses Kreuz, würde die Zahl 2 als Adresse erkannt werden und zu einem Adressfehler mit vier Bömbchen führen.

Nun zum SP, der hier vor der Operation um die Wortlänge verringert wird. Dies resultiert aus der Tatsache, daß der Strack beim ATARI zur kleineren Adresse hin wächst. Mit dem '-' vor der Klammer zeigen wir an, daß das Adressregister, das in der Klammer steht, vorher verringert werden muß. Die Klammer hingegen deutet dem Prozessor, daß er keine neue Adresse an die Stelle des Registers setzen soll, sondern erst bei SP-2 (durch das Minus bei Wortlänge verringert sich die Adresse nämlich um 2) die Adresse des Speichers feststellen muß. Dort, in diese Speicherzelle, wird dann unsere Zahl geschrieben. Nach dieser Operation steht der SP dann bei SP-2.

Um an das BS zu kommen, benützen wir beim Assembler sogenannte TRAPS (Fallen), die den Prozessor in eine Ausnahmesituation führen. Beim ATARI werden die Traps wie folgt belegt:

```
trap   #1    ; GEMDOS
trap   #2    ; GEM (AES und VDI)
trap   #13   ; BIOS
trap   #14   ; XBIOS
```

Findet der Prozessor einen Trap in dem Programm, benützt er die Trapnummer als Pointer für die anzuspringende Adresse und sichert PC und SR.

Nach dem Anspringen dieser Adresse holt sich das BS die Funktionsnummer und die eventuell mit übergebenen Parameter und führt die Routine aus. Nach der Ausführung übergibt uns das BS im Register d0 den entsprechenden Wert oder eine Fehlernummer, die immer negativ ist. Da wir unseren SP mit der Funktionsnummer erniedrigt haben, müssen wir jetzt nach der Ausführung für die korrekte Adresse des Stracks sorgen, denn sonst läuft er uns irgendwann wie der Schreibtisch über.

Da der SP durch unsere Wordoperation um zwei Adressen verringert wurde, müssen wir nun durch

```
addq.l   #2,sp
```

die alte Position wieder herstellen. Wir addieren also durch add.l zwei Adressen hinzu. Da es sich um eine Addition unter 8 handelt, können wir hier die schnelle Variante verwenden, die nun addq.l benannt wird. Gute Assembler, wie der GFA-ASSEMBLER, ersetzen add. automatisch durch addq., falls es sich anbietet. Eigentlich ist unser BS-Aufruf damit ja schon fertig, es fehlt nur noch die Adresse des Bildschirms, und die steht in d0.

```
movea.l   d0,a0
```

bewegt die in d0 stehende Adresse in das Adressregister a0. Falls der Rückgabewert einer Funktion keine Adresse ist, müssen wir

```
move.(b,w,l)   d0,Ziel
```

verwenden. Unser kompletter BS-Call sieht also so aus:

```
move.w    #2,-(sp)  ; Bildschirmadresse
                      feststellen
trap      #14       ; Betriebssystemaufruf
                      (XBIOS)
addq.l    #2,sp     ; Strack korrigieren
movea.l   d0,a0     ; und Adresse merken
```

Toll, oder? So einfach kann Assembler sein!

Jetzt haben wir schon die Adresse des Bildschirms in d0 stehen und sollten uns erst einmal über den Bildschirm unterhalten. Ihn können wir uns, wie übrigens den gesamten Speicher, als eine Reihe von Bits vorstellen, die komplett auf eine Schnur aufgezogen sind. Wollen wir nun den 111. Punkt (Pixel) in die 10. Reihe setzen, hangeln wir uns einfach neun Reihen à 640 Pixel und 111 Pixel an dieser Schnur vorwärts. Genau gesagt, wir gehen an die Adresse Bildschirm+5871 und setzen dort das Byte auf -1 = Schwarz. Wer genau aufgepaßt hat, kann jetzt schon seine erste eigene Befehlszeile schreiben:

```
move.b   #-1,d0
```

Gut! Erinnern wir uns aber mal an die Verarbeitungsbreite! Wenn wir alles in .b erledigen würden, kostet uns das nur Rechenzeit, also gehen wir auf Word oder Long, denn eine Kette können wir in 1er, 2er oder 4er Schritten durchwandern, wobei wir ja auf eine hohe Geschwindigkeit Wert legen wollen. Es kommt auch dasselbe Ergebnis heraus, ob wir vier Adressen nacheinander mit -1 belegen oder diese gleichzeitig füllen. Daher ist

```
move.l   #-1,d0
```

genau richtig und auch von der Zeit optimal. Unser Prozessor liebt übrigens Daten in den Registern; da ist er dann besonders schnell und flink. Daher wollen wir, soweit möglich, alle Daten in Registern lassen.

Unser Monochrome-Bildschirm hat in der hohen Auflösung 640x400 Pixel, die wir ansprechen können. Daher belegen wir nun die Register d1 und d2 mit diesen Werten, die wir jedoch von

0 aus abzählen und um eins verringert übergeben. Da wir ja mit Long arbeiten, muß der Wert auch noch durch vier geteilt werden, was jedoch der Assembler für uns tut.

```
move.w    #398/4,d2
move.w    #638/4,d1
```

Aber halt, normal wäre das doch 399 und 639! Klar, jedoch hat der 68000er einen Sprungbefehl, der sich solange wiederholt, bis die Zahl auf -1 gesetzt ist. Dieser dbra-Befehl zieht bei jedem Durchgang 1 vom Datenregister ab und springt bei positiven Zahlen an das angegebene Label zurück. Würden wir die Zahl nicht um 1 verringern, erhielten wir statt 400x640 Punkten genau 401x641 Punkte.

Nun haben wir also die Register belegt und können unsere ersten vier Punkte (Long) 'schwärzen'. Da die Adresse in a0 steht, können wir ohne Probleme zur Tat schreiten:

```
move.l    d0,(a0)+
```

Den Großteil dieser Befehlszeile kennen wir ja schon! Aber handelt es sich hier bei dem (a0)+ um einen Schreibfehler? Nein! Wir weisen dem Prozessor mit dem '+' an, die Adresse, die in a0 steht, nach der Verarbeitung um die Verarbeitungsbreite zu erhöhen! Genauso könnten wir

```
move.l    d0,(a0)   ; Muster eintragen
addq.l    #4,a0     ; Adresse erhöhen
```

schreiben. Aber der Prozessor macht das ja schon für uns, so daß wir Platz und Rechenzeit sparen, indem wir das Register nach der Verarbeitung automatisch erhöhen. Unser Programm sieht also erst mal so aus:

```
move.w    #2,-(sp)  ; Bildschirmadresse
                      feststellen
trap      #14       ; Betriebssystemaufruf
                      (XBIOS)
addq.l    #2,sp     ; Strack korrigieren
movea.l   d0,a0     ; und Adresse
                      merken
move.l    #-1,d0    ; Füllmuster
move.w    #398/4,d2 ; Koordinaten
move.w    #638/4,d1
move.l    d0,(a0)+  ; Punkt setzen
```

Tja, aber... Genau! Im Basic hatten wir doch zwei Schleifen! Die Schleifenbildung im Assembler ist ebenfalls sehr einfach. Benötigt werden für uns die Anzahl der Durchgänge, ein Label und ein Sprungbefehl. Die Durchgänge haben wir ja schon mit move.w #398/4,d2 festgelegt. Nur, wo wollen wir ein Label hinsetzen? Gehen wir doch noch mal zum Bildschirm zurück: Jede Zeile hat 640 Punkte. Und da wir ja 400 Zeilen füllen wollen, wäre es doch toll, wenn das Label bei move.w #638/4,d1 stehen würde. Oder? Richtig, zur ersten Schleife fehlt also nur noch der Sprungbefehl, und dieser lautet, wie schon angedeutet, dbra:

```
          move.w    #2,-(sp)
          trap      #14
          addq.l    #2,sp
          movea.l   d0,a0
          move.l    #-1,d0
          move.w    #398/4,d2
LOOP:     move.w    #638/4,d1
          move.l    d0,(a0)+
          dbra      d2,LOOP
```

Okay! Nun haben wir also die erste Schleife gelöst und setzen 400 Bildschirmpunkte. Aber da war doch noch eine zweite Schleife im Basic! Richtig! Um den ganzen Bildschirm zu setzen, benötigen wir auch noch eine zweite Schleife. Versuchen Sie doch mal, diese Schleife zu setzen; das nötige Wissen haben Sie ja jetzt schon. Alles klar? Wir starten die zweite Schleife bei move.l d0,(a0)+, denn diese Zeile muß bei jedem Durchgang der ersten Schleife 640 Bildpunkte setzen. Unser fast fertiges Listing sieht jetzt also so aus:

```
          move.w    #2,-(sp)  ; Bildschirmad-
                                resse feststellen
          trap      #14       ; Betriebssystem-
                                aufruf (XBIOS)
          addq.l    #2,sp     ; Strack korri
                                gieren
          movea.l   d0,a0     ; und Adresse
                                merken
          move.l    #-1,d0    ; Muster zum
                                "Bildfüllen"
          move.w    #398/4,d2 ; Vertikale Auf-
                                lösung
LOOP1:    move.w    #638/4,d0 ; Horizontale
                                Auflösung
LOOP2:    move.l    d0,(a0)+  ; Bildmuster ein-
                                tragen
          dbra      d1,LOOP2  ; zu Loop2 ver-
                                zweigen bis d1=-1
          dbra      d2,LOOP1  ; Zu Loop1 ver-
                                zweigen bis d2=-1
```

Um beim Basic zu bleiben: LOOP1 stellt die FOR NEXT Schleife für H% und LOOP2 die FOR NEXT Schleife für V% dar.

Dieses Listing wäre schon lauffähig, wenn wir am Ende unseres Programms dieses auch beenden würden. Auch hierfür stellt uns das Betriebssystem eine Funktion zur Verfügung. Nämlich PTERM vom GEMDOS.

```
clr.w   -(sp)  ; PTERM, Programm beenden
trap    #1     ; Betriebssystemaufruf (GEMDOS)
```

Mit clr.w 'übergeben' wir die Funktionsnummer 0 an den SP und rufen über trap #1 das Gemdos auf. Alles, was hinter dieser Funktion steht, wird nie erreicht, so daß wir den Strack auch nicht korrigieren müssen.

Unser fertiges Programm sieht also so aus:

```
Start:    move.w    #2,-(sp)  ; Bildschirmad-
                                resse feststellen
          trap      #14       ; Betriebssystem-
                                aufruf (XBIOS)
          addq.l    #2,sp     ; Strack korri-
                                gieren
          movea.l   d0,a0     ; und Adresse
                                merken
          move.l    #-1,d0    ; Muster zum
                                "Bildfüllen"
          move.w    #398/4,d2 ; Vertikale Auf-
                                lösung
LOOP1:    move.w    #638/4,d1 ; Horizontale Auf-
                                lösung
LOOP2:    move.l    d0,(a0)+  ; Bildmuster ein-
                                tragen
          dbra      d1,LOOP2  ; zu Loop2 ver-
                                zweigen bis d1=-1
          dbra      d2,LOOP1  ; Zu Loop1 ver-
                                zweigen bis d2=-1
          clr.w     -(sp)     ; PTERM, Pro-
                                gramm beenden
          trap      #1        ; Betriebssystem-
                                aufruf (GEMDOS)
          ; Hier kommt das Programm nie hin!
```

War doch gar nicht so schwer, oder?

Assembler ist wirklich sehr einfach zu erlernen, wenn man sich an einige Regeln hält und experimentierfreudig ist. Stören Sie sich bitte auch nicht daran, daß Sie von Zeit zu Zeit einen wahren Bombenhagel (Systemabsturz) erleben werden. Auch Programmierer, die das System sehr gut kennen, erleben hier und da faszinierende Abstürze.

Jetzt können Sie selbst mal loslegen. Verändern Sie doch mal die -1 bei move.l #-1,d0 in move.l #5,d0 und schauen Sie, was da kommt! Ihrer Phantasie soll hier keine Grenze gesetzt sein. Durch das #% Zeichen können Sie auch binäre Zahlen als Muster verwenden. Beachten Sie jedoch hierbei die Verarbeitungsbreite 32 Bit (32 Stellen), 16 Bit (16 Stellen) oder 8 Bit (8 Stellen)! Genauso gut können Sie aber auch im Hexadezimalsystem arbeiten, wenn Sie #$ verwenden.

Etwas zum Nachdenken bis zur nächsten Folge haben wir auch für Sie: Die Routine läßt sich noch weiter verbessern und optimieren. Probieren Sie es aus! Wir sind auf Ihre Ergebnisse gespannt und erwarten für die Zukunft viele in Assembler geschriebene Programme. In der nächsten Folge wollen wir uns mal mit der Textausgabe in Assemblerprogrammen beschäftigen und eine Befehlsübersicht für Sie darstellen. Auf unserer Leserdiskette finden Sie die komplette Befehlsübersicht, den gesamten Kurs sowie ein paar Listings für den GFA-ASSEMBLER.

(rk)

Typen, Trends und schönes Wetter

Die European Computer Trade Show '90 in London

Vom 1. bis 3. April öffnete bei strahlendem Sonnenschein das Business Design Centre im Herzen Londons seine Pforten. Die European Computer Trade Show stand diesmal voll und ganz im Zeichen des heraneilenden Binnenmarktes.

Gemäß des englischen Marktes konnte man für die Amigas und Ataris fast nur Spiele bewundern. Auch nur für den STE, der auf der Insel weit mehr verbreitet ist als bei uns, gab es schon eigene Spiele. Nach Angaben der Softwareproduzenten gibt es jedoch weit mehr Probleme mit der Kompatibilität zwischen dem alten und dem erweiterten ST als bisher angenommen. In England sind daher die meisten Spiele mit einem Aufkleber versehen, der sie als STE tauglich bzw. untauglich kennzeichnet. Einige der neuen Programme laden auch je nach verwendetem Modell unterschiedliche Datas, um die besseren Möglichkeiten des STE wenigstens teilweise auszunutzen.

In Grobritannien ist der STE übrigens auch in der 520KB Version erhältlich. Der 520STE mit eingebautem Laufwerk kostet zusammen mit einem Spiel und Joystick 269 Pfund, also etwa 770 DM. Da der ST in England hauptsächlich für Spiele eingesetzt wird, gibt es auch wenige Programme für die bei uns üblichen monochromen Monitore. Etwa 85% der ST Anwender besitzen einen Farbmonitor, in Deutschland kaum ein Drittel.

Ganz überraschend fand ich jedoch ein DTP Programm für etwa 60 (in Worten: sechzig) DM. Das Programm ist zwar teilweise relativ langsam, die Ergebnisse lassen sich aber durchaus sehen. Es läuft ab 1 MB und unterstützt bisher alle gängigen 9 und 24 Nadeldrucker. Treiber für Laserdrucker sollen bis Ende Mai folgen. Ob es auch in Deutschland erhältlich sein wird, ist bisher nicht bekannt.

Doch nun zurück zur Messe. Natürlich gab es rechtzeitig zur Fußball-Weltmeisterschaft in Italien jede Menge Fußballsimulatoren. Auch als Manager unterschiedlicher Teams kann man sich einmal mehr versuchen. Die wohl eindrucksvollste Version eines Fußballspiels lief auf einem Archimedes, der natürlich jede Menge Power für eine fließende Animation bot.

Auch Ballerspiele mit Lichtpistolen sind im Kommen. Besonders preiswerte Lichtpistolen gab es für den Amiga (ab etwa 90 DM).

Auch Strategiespiele erfreuten sich großer Beliebtheit. So kann man beispielsweise als Hannibal, Caesar, Alexander der Große oder auch als Napoleon im großen Topf der Geschichte herumrühren.

Weit interessanter als die neuen Spiele war die Gerüchteküche. Die ECTS ist eine reine Fachmesse, der Eintritt ist auch nur für Fachleute möglich. Kurz: die ideale Umgebung für allerhand inoffizielle Ankündigungen.

Wie auch schon auf der CeBIT war der zu erwartende Atari Portfolio auf 80286er Basis in aller Munde. Doch auch verschiedene Softwareentwickler und Mitarbeiter von Atari UK wollen von einem ST kompatiblen Taschencomputer gehört haben. Ob man dann endlich unter der Schulbank DTP betreiben kann, konnte nicht in Erfahrung gebracht werden, uninteressant wäre jedoch ein Portfolio ST sicherlich nicht.

Ein anderes Gerücht, diesmal von einem höheren Mitarbeiter von Commodore, will wissen, daß Irving Gold, Geschäftsführer von Commodore, eine enge Zusammenarbeit mit der Firma NEXT anstrebt. Für Steve Jobs; Mitbegründer der Firma APPLE und derzeitiger Leiter von NEXT, käme ein solcher Zusammenschluß sicher nicht ungelegen.

Über den Amiga 3000, der ja schon vergeblich auf der CeBIT gesucht wurde, waren zwar die abenteuerlichsten Dinge zu hören, etwas Handfestes gab es jedoch nicht. Möglicherweise wird man ihn auf der Which Computer? Show zu Gesicht bekommen. Commodore hat für diese Messe etwas Besonderes angekündigt!?

Bereits erhältlich ist aber ein Erweiterungskit, den die Firma Pheonix für alle Amiga 1000 Benutzer herausgebracht hat. Das Pheonix Board ersetzt das alte Amiga Mother Board und alle Erweiterungskarten, Kontroller etc. bleiben jedoch erhalten. Die Erweiterung kostet 480 Australische Dollar (640 DM) und soll die Leistung des A1000 auf das Niveau des A2000 emporstoßen. Die Platine ist in der Grundausstattung mit 1 MB bestückt, kann jedoch auf 2 MB (10 MB extern) aufgerüstet werden. Sie bietet auch Steckplätze für den neuen Chipsatz (ECS), über den man derzeit ja überall munkelt.

Zusammengefaßt waren in London zwar keine Sensationen zu finden. Doch allein die vielen Gerüchte und die Ruhe, die man wohl auf keiner anderen Computermesse findet, waren die Reise wert.

Martin Wollny

GFA BASIC Tips & Tricks

DIN A4 Grafiken mit neuer Füll-routine

Wer in der Clubzeitung 5/89 auf Seite 12 den Artikel "Doppelt ist größer" gelesen hat, wird festgestellt haben, daß dazu offensichtlich das Listing fehlt. Dies soll hier nachgeholt werden. Zusätzlich aber wird hier eine neue Füllroutine vorgestellt. Der gewöhnliche FILL-Befehl des GFA-BASIC macht nämlich bei den aktuellen XBIOS(3)-Bildschirmrändern halt - dies ist für ein zweiscreeniges Bild aber tödlich (oder zumindest störend).

Zunächst aber zum Einrichten der Doppelscreen (PROCEDURE install.bild): Der Basicspeicher wird durch RESERVE um 65000 Bytes verringert. Hinter der neuen HIMEM-Adresse kommt an die nächste, durch 256 teilbare Adresse (wichtig für XBIOS(5)!), der neue Bildschirmbereich, der nun 64000 Bytes groß ist. Dort wird eine kleine Beispielgrafik erstellt und mit XBIOS(5) die obere Hälfte zunächst als neuer Bildschirmspeicher angemeldet.

In der PROCEDURE main.prg kann durch die Cursortasten "nach oben/ unten" das Bild gescrollt werden. Dabei wird von der aktuellen Adresse des sichtbaren Ausschnitts das kleinste gemeinsame Vielfache von 80 und 256 - namentlich 1280 - abgezogen bzw. dazugezählt und der neue Ausschnitt mittels XBIOS(5) angemeldet. Mit der Return-Taste wird das Programm beendet und RESERVE und XBIOS(5) auf die Startwerte gebracht.

Außerdem kann im Hauptprogramm durch Druck auf die linke Maustaste ein gewünschter Screenbereich ausgefüllt werden. Dabei wird zuerst die eine Bildhälfte in einem gesonderten Screenbereich mit schwarzem Füllmuster ausgefüllt. Indem dann die Originalhälfte im XOR-Modus darüberkopiert wird, "stanzt" man somit die Maske aus, d.h. die Umrisse und die übrige Grafik werden entfernt. In der Arbeitsscreen steht somit nur die Mustermaske.

Nun wird diejenige Pixelzeile untersucht, die an die andere Hälfte angrenzt - also 0 bzw. 399. Wo sich hier gesetzte Pixel finden, sollte das Muster die andere Bildhälfte weiterfüllen. Da das der FILL-Befehl aber nicht bringt, merkt man sich einfach diese Stellen, um den Füllbefehl im nächsten Durchgang auf diese Punkte anzusetzen. Sodann wird die Maske mit dem gewünschten Füllmuster gefüllt und aus dem Arbeitsbereich in den Bildspeicher zurückkopiert. Nun wird, wie schon angedeutet, die andere Screenhälfte in den Arbeitsbereich kopiert und an den ermittelten X-Stellen - Y = 399 bzw. 0 - wieder erst mal mit der Maske gefüllt, usw.

Diese Schleife wird solange durchlaufen, bis die Maske nicht mehr an die andere Hälfte anstößt. Anzumerken wäre noch, daß das Füllen ein klein wenig dauert (klar), sich aber während dieser Zeit nichts auf dem Bildschirm tut. Also nicht gleich Break oder Reset drücken! Außerdem das Programm immer mit Return verlassen, damit Bildschirm- und Basicspeicher wiederhergestellt werden können.

Andreas Walthes

```
@install.bild   !* Doppelscreen mit Scolling und Füllroutine *
@main.prg       !* v. A. Walthes, Leidlstrae 3, 8390 Passau *
@uninstall      !* GFA-BASIC 2.0/3.0 (?) 14.01.1990 ATARI ST *
'===================================================
DEFFN adr.bld=(VARPTR(bld%(0))+255) AND &HFFFFF00 !
                     Adresse des Füll-Arbeitsspeichers als
'                    Funktion!
DEFFN setscr(l.ad%,p.ad%)=XBIOS(5,L:l.ad%,L:p.ad%,-1)  !
                                     XBIOS(5)-Funktion
'---------------------------------------------------
PROCEDURE install.bild
  CLS
  start.res%=FRE(0)             ! Startwerte sichem
start.scr%=XBIOS(2)
  RESERVE FRE(0)-65000          ! Platz schaffen für 2 Screens
  screen.adr.1%=(INT((HIMEM+2)/256)+1)*256  ! Adresse der
  '                                          ersten Hälfte
   VOID @setscr(screen.adr.1%+16000,screen.adr.1%) ! xbios-
   '                                     Adressen verändern
  BMOVE start.scr%,XBIOS(2),32000 ! Neuen Screenbereich
  '                                 löschen
  BMOVE start.scr%,XBIOS(2)+32000,32000
  REPEAT                                    ! Kleine Grafik
    INC b%
    DRAW  TO RND*640,RND*800-200
  UNTIL b%=40
  VOID @setscr(screen.adr.1%,screen.adr.1%) ! xbios-Adressen
  '                                              verändern
  DIM q%(5),z%(5),r%(8)             ! BITBLT-Parameter
  q%(1)=640                         ! (Bei GFA-BASIC 3.0 anders?)
  q%(2)=400
  q%(3)=q%(1)/16
  q%(5)=1
z%(1)=640
  z%(2)=400
z%(3)=q%(3)
  z%(5)=1
  r%(2)=639
r%(3)=399
r%(6)=639
r%(7)=399
RETURN
'
```

```
PROCEDURE main.prg
REPEAT
  REPEAT
    i$=INKEY$
    MOUSE x%,y%,k%
    IF k%=1                    ! Füllen ?
      @new.fill(x%,y%,2,4)     ! Koordinaten und Füllmuster über-
      '                          geben.
    ENDIF
  UNTIL i$>""
  HIDEM
  IF i$=CHR$(0)+CHR$(72) AND offset%>=1280   ! Nach oben
    '                                           scrollen
    SUB offset%,1280
    VOID @setscr(screen.adr.1%+offset%,screen.adr.1%+offset%)
  ENDIF
  IF i$=CHR$(0)+CHR$(80) AND offset%<=30720   ! Nach unten
    '                                           scrollen
    ADD offset%,1280
    VOID @setscr(screen.adr.1%+offset%,screen.adr.1%+offset%)
  ENDIF
  SHOWM
UNTIL i$=CHR$(13)          ! Return = Ende
RETURN
'
```

```
PROCEDURE uninstall
  VOID @setscr(start.scr%,start.scr%)
  RESERVE start.res%
  EDIT
RETURN
'
```

```
PROCEDURE new.fill(x%,y%,f1%,f2%)
  LOCAL cv%,cb%,gn$,ir%,jr%,ik%,ofst%,erste!,xx$,pe%,as%,gn%
  '
  cv%=XBIOS(3)                ! Aktuelle Bildschirmadressen sichern
  cb%=XBIOS(2)
  DIM bld%(8190)              ! Speicher für Zusatzscreen reser-
  '                             vieren
  '
  IF y%+(offset%/80)<=399     ! Wenn Füllpunkt in der ersten Bild-
    '                           hälfte,
    ofst%=32000               ! dann Offset-Startwert 32000
    '                         ! (wird später mit 0 vertauscht)
  ELSE                        ! Ansonsten den zwischen 400 und
    '                           799 liegenden Y%-Wert an
    SUB y%,400                ! 0-399 Px anpassen.
  ENDIF
  '
  DO
    ofst%=ofst% XOR 32000     ! Bildhälftenadressen-Offset ver-
    '                           tauschen
    BMOVE screen.adr.1%+ofst%,@adr.bld,32000  ! Bildhälfte in
    '                                           Arbeitsscreen
    VOID @setscr(@adr.bld,XBIOS(2))   ! moven und
    '                                   XBIOS(3)anpassen.
    DEFFILL ,1,1              ! Muster der Maske (Schwarz)
    luecke!=-1
    '
    IF erste!=0               ! Wenn dies erster Schleifendurch-
      '                         gang ist...
      FILL x%,y%+(offset%/80)   ! ...von Startpunkt füllen
    ELSE                      ! Sonst das an den Hälftengrenzen evtl.
      FOR ir%=1 TO 80         ! durchgebrochene Muster vom letzten
        '                     ! Durchgang weiterlaufen lassen.
        ik%=ASC(MID$(gn$,ir%,1))
        IF ik%                ! Im Flagstring Gn$ aktuelles Byte als
          '                   ! Durchbruchstelle vermerkt?
          FOR jr%=0 TO 7
            '                 ! Dann genaue Bits suchen und dort
            '                 ! füllen, falls noch nicht geschehen
            IF (ik% AND (2^(7-jr%))) AND
POINT((ir%-1)*8+jr%,(-399)*((ofst%/32000)-1))=0
              IF luecke!=-1
                CLR luecke!
                xx$=xx$+MKI$((ir%-1)*8+jr%)  ! Speichern, wo später Maske
                '                             ! noch gefüllt werden muß
              ENDIF
              FILL (ir%-1)*8+jr%,(-399)*((ofst%/32000)-1)
            ELSE
              luecke!=TRUE
            ENDIF
          NEXT jr%
        ELSE
          luecke!=TRUE
        ENDIF
      NEXT ir%
    ENDIF
    '
    r%(8)=6                   ! Maske in Arbeitsscreen "ausstanzen"
    q%(0)=screen.adr.1%+ofst%
    z%(0)=@adr.bld
    BITBLT q%(),z%(),r%()
    '
    IF gn$=""                 ! Wenn erster Durchlauf...
      gn$=SPACE$(160)         ! ...Bildzeile an Hälftengrenze holen.
      BMOVE 80*(-399)*((ofst%/32000)-1)+@adr.bld,VARPTR(gn$)
      +80,80
    ELSE                      ! Sonst...
      FOR a%=1 TO 80          ! ...auch holen, aber evtl. schon
        '                     ! bearbeitete X-Werte ausmaskieren
        pe%=PEEK(@adr.bld-1+a%-80*399*((ofst%/32000)-1))
        as%=ASC(MID$(gn$,a%,1))
        gn%=(as% XOR pe%) AND pe%
        gn$=gn$+CHR$(gn%)
      NEXT a%
    ENDIF
    gn$=MID$(gn$,81)
    '
    DEFFILL ,f1%,f2%          ! Gewünschtes Füllmuster einstellen
    IF erste!=0               ! Wenn erster Durchlauf...
      erste!=-1
      FILL x%,y%+(offset%/80)   ! ...An Startpunkt füllen
    ELSE                      ! Sonst...
      IF xx$>""               ! ...An den bei der Maskenerstel-
lung
        FOR ir%=1 TO LEN(xx$)/2 !    ermittelten Stellen füllen.
          FILL CVI(MID$(xx$,ir%*2-1,2)),(-399)*((ofst%/32000)-1)
        NEXT ir%
      ENDIF
    ENDIF
    '
    q%(0)=@adr.bld              ! Das in der Arbeitsscreen stehende
    z%(0)=screen.adr.1%+ofst%   ! Füllmuster auf das Original-
    '                             transparent draufkopieren
    r%(8)=7
    BITBLT q%(),z%(),r%()
    '
    xx$=""
    EXIT IF gn$=STRING$(80,0)  ! Wenn Maske nicht an Hälfte-
  LOOP                         ! grenze gestoßen ist, ENDE
  VOID @setscr(cv%,cb%)
  ERASE bld%()
RETURN
'
```

Jedem seinen Schlüssel

Haben auch Sie Daten, die nicht gleich jeder sehen sollte? Dann haben Sie sich sicher auch schon mal mit der Datenverschlüsselung beschäftigt. Die einfachste Möglichkeit, die Daten zu verschlüsseln, besteht darin, zu jedem Byte z.B. eine Konstante zu addieren.

Kompilizierter wird es, wenn dem Alphabet ein anderes zusammengewürfeltes Alphabet zugeordnet wird. In beiden Fällen wird jedem Buchstaben ein definitiver Wert zugeordnet. Für das Entschlüsselungssystem ist dies keine schwierige Aufgabe. Anders sieht es aus, wenn die Bytes mit immer wechselnden Zahlen addiert werden. Denn hier weist ein Buchstabe immer andere Werte nach der Verschlüsselung auf: z.B. Code = 1234

Text:

Dies ist ein zu verschlüsselnder Text

Verschlüsselung:

1234123412341234123412341234123412341

Resultat:

ekhw!kvx!glr!\x$wguwdjotuhpmfhv!Vh\u

Nun ist es möglich, daß ein Zeichen aus dem Quelltext unterschiedlich verschlüsselt werden kann. Die Entschlüsselung ist dann allerdings nicht mehr so einfach. Wird der Text mit dem Code durch Addition verknüpft, so ist der Code bei der Dechiffrierung vom Text zu subtrahieren. Wird der Text jedoch mit dem Code durch ein logisches XOR verknüpft, so kann man bei der Entschlüsselung denselben Algorithmus verwenden, denn durch eine zweimalige XOR-Verknüpfung erhält man wieder das Orginal.

Eine einfache und schnelle Möglichkeit Daten mit XOR zu verschlüsseln und zu entschlüsseln, geben uns die Grafikbefehle. Und damit ist eine neue Verschlüsselungsart geboren, die sog. "grafische Verschlüsselung".

Die zu verschlüsselnde Datei wird in den Bildschirm geladen und grafisch XOR mit einem Bild, mit Strichen oder Füllmustern ausgestattet. Anschließend wird sie abgespeichert. Zur Entschlüsselung ist die ganze Prozedur nun noch einmal auszuführen. Will man dem Ganzen noch einen Profi-Touch geben, so schaltet man vor den grafischen Spielereien in einen Hintergrundbildschirm.

Eduard Rhode

```
'
' .von E.Rhode........ '
' .Neu Radenbeck 23... '
' .2121 Thomasburg.... '
'
'          Jedem seinen Schlüssel
'          ----------------------
'
' Verschlüssellungsprogramm
'
PROCEDURE schluessel(code$,file$)
' code$= code   file$ = filename der zu verschlüsselnden Datei
  LOCAL c$,f$,s$,s%               ! Local-Definitionen
  IF EXIST(file$)                 ! Wenn File existiert,
    FOR i&=1 TO 32/LEN(code$)+1   ! code$ auf min. 32 Zeichen erweitern.
      c$=c$+code$
    NEXT i&
    code$=LEFT$(c$,32)            ! Code$ auf 32 Zeichen zurechtschneiden.
    FOR i&=1 TO 16                ! Umwandlung des Codes in ein Füllmuster.
      f$=f$+MID$(code$,(i&-1)*2,2)
    NEXT i&
    DEFFILL 1,f$                  ! Füllmuster definieren.
    GRAPHMODE 3                   ! Zeichenmodus definieren.
    OPEN "u",#1,file$             ! File öffnen.
    s%=LOF(#1)                    ! Länge des Files bestimmen.
    WHILE s%>32000
      BGET #1,XBIOS(3),32000      ! Daten von File holen auf Bildschirm.
  PBOX 0,0,639,399                ! Gefülltes Rechteck zeichnen.
      RELSEEK #1,-32000           ! Filepointer um 32000 zurücksetzen.
      BPUT #1,XBIOS(3),32000      ! Bildschirm in File schreiben.
      SUB s%,32000
    WEND
    BGET #1,XBIOS(3),s%           ! Restliche Bytes holen.
    PBOX 0,0,639,399              ! Rest siehe oben.
    RELSEEK #1,-s%
    BPUT #1,XBIOS(3),s%
    CLOSE #1                      ! File wieder schliessen.
    CLS                           ! Bildchirm löschen.
    DEFFILL 1,1                   ! Füllmuster zurücksetzen.
    GRAPHMODE 1                   ! Zeichenmodus zurücksetzen.
  ENDIF
RETURN
```

DATA-Wüsten werden zu DATA-Oasen

Datazeilen enthalten Daten. Das Abschreiben dieser Datazeilen artet meist in eine mühsame stupide Tipparbeit aus, die man so gering wie möglich halten sollte. Außerdem wäre eine übersichtliche Vorlage erstrebenswert. Um unnötigen Ärger zu ersparen, sollte die "DATA-Wüste" zusätzlich Checksummen enthalten.

DATA-Wüsten werden zu DATA-Oasen durch folgende drei kurze Programme: ein Programm für den Programmierer zum Generieren von Datazeilen aus Files, ein Programm zum Einlesen der Daten und ein Programm, um sich die Tipperei zu vereinfachen. Der Programmcode ist kurz und gut dokumentiert.

Die Bedienung:

Data_gen.gfa: Ein Programm zum Generieren von Data-Zeilen aus einem File

Drei Parameter müssen übergeben werden: der Filename des zu wandelnden Files, der Name der zu kreierenden Datei und ein Flag, ob der Code mit oder ohne Checksumme erzeugt werden soll. Es werden dann Datazeilen mit 16 Hexdaten und gegebenenfalls dreistelliger Checksumme erzeugt.

Data_ein.lst: Eine Procedure zur manuellen Eingabe von Datazeilen

Bei Bytegröße sind die meisten hexadezimalen Zahlen zweistellig, die Dezimalen schon dreistellig. Leider besitzt der ST keinen Hex-Block zur Eingabe, wobei der Ziffernblock von der Anzahl der Tasten her schon ausreichen würde. Daher wurden einfach Tasten umbenannt. Die obere linke Taste "A" = B, "/" = C, "*" = D, "-" = E und "+" = F. Die "."-Taste wurde nicht umdefiniert, da keine Kommas eingegeben werden brauchen. So kann man sich noch ein paar Tastendrücke sparen.

Eigentlich sollte es keine Datazeilen ohne Checksumme geben. Trotzdem verzichten manche Programmierer auf diesen Komfort. Ich gebe lieber drei Ziffern mehr ein, als dann stundenlang Fehler zu suchen. Nach dem 16. Datum kann eine dreistellige Checksumme eingegeben werden. Füllen Sie bitte die ersten Stellen gegebenenfalls mit Nullen auf. Beim Eingabeprogramm wurde auf eine Überprüfung der Checksumme verzichtet. Das Programm wäre zu groß geworden. Die Checksumme kann aber mit eingege-

```
'
PRINT "   Eduard Rhode,  Neu Radenbeck 23,   2121 Thomasburg"
PRINT
PRINT "   Data Zeilen Generator"
ALERT 1,"Mit Checksumme",1,"Ja|Nein",check|
CLS
PRINT "Welche Datei soll gewandelt werden?"
DO
 FILESELECT "\*.*",a$,b$
 EXIT IF b$""
LOOP
CLS
PRINT "Wie soll die neue Datei heißen?"
DO
 FILESELECT "\*.*",a$,z$
 EXIT IF b$""
LOOP
OPEN "i",#1,b$                          ! Eingabedatei öffnen
OPEN "o",#2,z$                          ! Ausgabedatei öffnen
DO                                      ! Endlosschleife Anfang
c$="Data"                               ! Speicherstring belegen oder zurücksetzen
 FOR i&=1 TO 16                         ! 16 Daten pro Zeile
  EXIT IF EOF(#1)                       ! Abbruch, falls Dateiende erreicht
  a|=INP(#1)                            ! ein Byte holen
  c$=c$+","+HEX$(a|,2)                  ! Speicherstring, um Komma und
  '                                     ! Hexzahl von Byte erweitern
  ADD check%,a|                         ! Checksumme bilden
 NEXT i&                                ! nächstes Byte
 IF check|                              ! Wenn Checksumme gebildet werden soll,
  c$=c$+","+HEX$(check%,3)              ! Speicherstring um Checksumme erweitern
  check%=0                              ! Checksumme zurücksetzen
 ENDIF
 MID$(c$,5,1)=" "                       ! Erstes Komma im Speicherstring löschen
 PRINT #2,c$                            ! Speicherstring ausgeben
 EXIT IF EOF(#1)                        ! Ende, wenn Dateiende erreicht
LOOP                                    ! Endlosscheife Ende
CLOSE                                   ! alle Dateien schliessen

' data_ein.gfa
'
' von E.Rhode
' Neu Radenbeck 23
' 2121 Thomasburg
'
PROCEDURE data_ein_init
 ' Funktionstaste belegen
 KEYDEF 10,"DATA "                      ! F10 belegen
 ' daten lesen
 DIM data%(1,17)                        ! Datenfeld definieren
 RESTORE scandatas                      ! Datenzeiger auf Scandatas richten
 FOR i%=0 TO 17                         ! Scancode lesen
  READ a$                               ! Datum lesen
  LET data%(0,i%)=VAL("&H"+a$)          ! Datum umwandeln und speichern
 NEXT i%                                ! nächstes Datum
 RESTORE asciidatas                     ! Datenzeiger auf ASCII-Daten richten
 FOR i%=0 TO 17                         ! ASCII-Code lesen
  READ a$                               ! Datum lesen
  LET data%(1,i%)=VAL("&H"+a$)          ! Datum umwandeln und speichern
 NEXT i%                                ! nächstes Datum
 aa%=XBIOS(16,L:-1,L:-1,L:-1)           ! Vektor auf Tastaturtabelle erfragen
 ret%=LPEEK(aa%)                        ! Vektor auf Unshift-Tabelle erfragen
 RESERVE 100000                         ! Speicher freigeben
 r%=MALLOC(256)                         ! Speicher reservieren

 ' tabelle copieren
 FOR i%=0 TO 31                         ! 32*4=128 Bytes kopieren
  LPOKE r%+i%*4,LPEEK(ret%+i%*4)        ! 4 Bytes lesen und schreiben
 NEXT i%                                ! nächstes Langwort
```

ben werden und wird dann beim Lesen der Datas überprüft. Nach Drücken der Return-Taste wird die letzte Zeile abgespeichert und automatisch ein "DATA" ausgegeben, sodaß dieser Befehl auch nicht mehr geschrieben werden muß.

Das Programm wird mit der Esc-Taste verlassen. Die letzte Zeile wird nicht mit abgespeichert. Daher drücken Sie vor Esc noch einmal Return. Die Datei wird geschlossen und kann jetzt hinzugemerget werden.

Dataread.lst: Datei mit Procedure zum Einlesen und Speichern der Daten

Zuerst stellt man mittels der RESTORE-Marke den Datenlesezeiger auf den richtigen Anfang. Sodann kann die Prozedure dataslesen (par1, par2,par3) aufgerufen werden. Ihr müssen drei Parameter übergeben werden. Der erste enthält die Anzahl der zu lesenden Daten. Der zweite enthält eine Adresse, an der die Daten transportiert werden sollen, und der letzte zeigt an, ob mit (=-1) oder ohne (=0) Checksumme gelesen wird. Erfolgt ein Checksummenfehler, so wird eine Fehlermeldung mit der fehlerhaften Datenzeilennummer ausgegeben. Gerechnet wird hier ab der Marke, auf die der Restore-Zeiger gerichtet wurde.

Eduard Rhode

```
  ' tabelle ändern
  FOR i%=0 TO 17                              ! 18 Daten ändern
    POKE r%+data%(0,i%),data%(1,i%)           ! Datum ändern
  NEXT i%                                     ! nächstes Datum
  aa%=XBIOS(16,L:r%,L:-1,L:-1)                ! neue Tabelle dem System mitteilen
RETURN
PROCEDURE data_ein_ende
  aa%=XBIOS(16,L:ret%,L:-1,L:-1)              ! alte Tabelle wieder aktuell
  v%=MFREE(r%)                                ! rerservierten Speicher wieder freigeben
RESERVE                                       ! Speicherplatz wieder GFA-BASIC zuteilen
RETURN
PROCEDURE data_ein_datas
  scandatas:                                  ! Datenmarke
  DATA 63,64,65,66
  DATA 67,68,69,4A
  DATA 6A,6B,6C,4E
  DATA 6D,6E,6F,72
  DATA 70,71
  asciidatas:                                 ! Datenmarke
  DATA 31,32,33,34
  DATA 35,36,37,38
  DATA 39,41,42,43
  DATA 44,45,46,0D
  DATA 30,2C
RETURN

' Procedure zum Lesen der Daten
'
PROCEDURE dataslesen(anzahl%,wohin%,check&)
  ' anzahl% der zu lesenden bytes
  ' wohin mit den daten
  ' check&   1 mit checksumme    0 ohne checksumme
  DO                                          ! Endlosschleife Anfang
    INC z%                                    ! Zeilenzähler +1
    FOR i&=1 TO 16+check&                     ! Daten pro Zeile
      EXIT IF j%=anzahl%+check&               ! Ausgang, wenn alle Daten gelesen
      READ a$                                 ! Daten lesen
      EXIT IF LEN(a$)=3                       ! Ausgang, wenn Checksumme gelesen
      INC j%                                  ! Datenzähler +1
      a%=VAL("&H"+a$)                         ! Umwandeln der Daten in dezimal
      ADD c%,a%                               ! Checksumme bilden
      POKE wohin%+j%-1,a%                     ! Gelesene Daten abspeichern
    NEXT i&                                   ! nächstes Datum
    IF check&=1                               ! wenn Checksumme
      IF c%VAL("&H"+a$)                       ! Überprüfen, ob errechnete gleich gelesener Summe
        ALERT 3,"Fehler in Datazeile|"+STR$(z%),1,"Ende",al&  <+><+><+><+><+><+>    ! Fehlermeldung
        END                                   ! wenn Fehler Ende
      ELSE                                    ! sonst
        c%=0                                  ! Checksumme zurücksetzen
      ENDIF
    ENDIF
    EXIT IF j%=anzahl%                        ! Ausgang, wenn alle Daten gelesen
  LOOP                                        ! Schleifenende
RETURN                                        ! Procedureende = zurück
```

GFA-CLUB-Händler

- HABA COMPUTER AG
Münsterstr. 9
2000 Hamburg 54
0 40 / 5 66 01-1

- Sienknecht Bürokommunikation
Heiligengeiststr. 20
2120 Lüneburg
0 41 31 / 4 61 22

- EDV-Beratungscenter
Cuxhavener Str. 1
2170 Hemmoor
0 47 71 / 74 04

- Der Computerladen
Coriansberg 2
2210 Itzehoe

- Layout Service
Eckernförder Str. 83
2300 Kiel 1
04 31 / 18 09 75

- Computer-Studio oHG
Am Lornsenpark 31
2380 Schleswig
0 46 21 / 2 98 51

- Computersysteme
Am Pferdemarkt 5
2720 Rotenburg / W.
0 42 61 / 21 29

- PZ-Sequenzdesign GBR
Bahnhofstr. 24
2842 Lohne (Oldb.)
0 44 42 / 58 64

- IVEMA TELEPOINT
Posthalterweg 4a
2900 Oldenburg

- Carl Wöltje GmbH
Heiligengeiststr. 6
2900 Oldenburg
0441/40 45 93

- Friesland Data
Berliner Str. 17
2948 Schortens

- Com Data
Am Schiffergraben 19
3000 Hannover 1
05 11 / 32 67 36

- SMC GmbH
Schulstr. 7
3007 Gehrden 1
0 51 08 / 51 67

- F & T Computervertrieb
Am Hornberg 1
3040 Soltau
0 51 91 / 1 65 22

- Witte Bürotechnik
Koopmannshof 69
3250 Hameln 1
0 51 51 / 75 95-96

- Dr. Hildebrandt & Buchholz
Magdeburger Kamp 10
3380 Goslar
0 53 21 / 8 07 31-32

- TRIFOLIUM
Grassweg 14
3500 Kassel
05 61 / 282824

- OFFICE AUTOMATION
Van der Werff Str. 6
4000 Düsseldorf 1
02 11 / 77 20 31

- Computer Hard- & Software
Irenenstr. 76 c
4000 Düsseldorf-Unterrath

- HOCO EDV-Anlagen GmbH
Ellerstr. 155
4000 Düsseldorf 1
02 11 / 78 52 13-14

- Rota Datenverarbeitung GmbH
Süchtelner Str. 7
4060 Viersen 1
0 21 62 / 3 10 11

- ssb software-studio brakmann
Württembergstr. 34
4200 Oberhausen 11
02 08 / 66 12 95

- Bottroper Computer Team GbR
Pestalozzistr. 35
4250 Bottrop
0 20 41 / 6 21 20

- R. Wischolek Computertechnik
Mesteroth 9
4250 Bottrop-Feldhausen
0 20 45 / 8 16 38

- EDV-Thiel
Lörhof 8
4350 Reckling-hausen
0 23 61 / 2 80 29

- OCB GmbH
Wallstr. 3
4422 Ahaus
0 25 61 / 50 21

- DELO-COMPUTER-TECHNIK
Kranenbusch 28
4600 Dortmund 15

- Mentis GmbH
Poststr. 15
4650 Gelsenkirchen
02 09 / 5 25 72

- CSF Computer & Software GmbH
Heeperstr. 106-108
4800 Bielefeld 1
05 21 / 6 16 63

- Knoll Computer
Wilhelmstr. 5
4904 Enger
0 52 24 / 44 19

- ProElektronic
Königstr. 17 - 19
4972 Löhne 1
0 57 32 / 10 19-0

- Rothkegel & Rolf
Vogelsanger Str. 278
5000 Köln 30
02 21 / 54 10 36

- MP Software Service
Reuterstr. 49
5060 Bergisch Gladbach 2
0 22 02 / 2 17 84

- RODA-SOFT
Bahnhofstr. 6
5120 Herzogenrath
0 24 06 / 7 91 00

- EDV-Vertrieb Gerhard-Hauptmann-Str. 33
5270 Gummersbach-Dieringhausen
0 22 61 / 7 46 60

- Bürocenter Lehr GmbH
Güterstr. 82
5500 Trier
06 51 / 20 97 10

- Computer Finke
Kipdorf 22
5600 Wuppertal 1
0202/45 32 33

- Wrede
Ruhrplatz 7
5778 Meschede
02 91 / 60 94

- W. Kräling KG
Pf 2240
5788 Winterberg
0283/504-8

- UTS - COMPUTER-DIENST
Langgasse 40
6082 Moerfelden-Waldorf
0 61 05 / 2 37 65

- Georg Heim oHG
Heidelberger Landstr. 194
6100 Darmstadt-Eberstadt
0 61 51 / 5 60 57

- Digitron GbR
Rodheimer Str. 34
6300 Gießen
06 41 / 8 55 66

- MCA Computer Center
Sindelfinger Allee 1
7030 Böblingen
0 70 31 / 22 60 15

- Böhmer Electronic GmbH
Wilhelm-Zapf-Str. 9
7080 Aalen
0 73 61 / 6 26 86

- Computerstudio W. Brock GmbH
Federnseestr. 17
7410 Reutlingen
0 71 21 / 3 42 87

- Dorfschmid Bürotechnik
Schulweg 18
7442 Neuffen /N.
0 70 25 / 64 42

- R. Gärtig Software-Entwicklung
Ringstr. 4
7450 Hechingen-Beuren
0 74 77 / 81 58

- SRE Computercenter
Schloßplatz 3
7450 Hechingen
0 74 71 / 1 45 07

- Martin & Martin Soft- und Hardware-vertrieb
Kruttenaustr. 9
7630 Lahr/Schw.
07821/4890 und 41050

- Kaltenbach EDV
Bachstr. 73
7465 Geislingen
0 74 33 / 52 44

- U. Meier Computersysteme
Ringstr. 4
7700 Singen /Htwl.
0 77 31 / 6 82 22

- Computer Fachgeschäft Rösler
Rheingutstr. 1
7750 Konstanz
0 75 31 / 2 18 32

- Computer und Zubehör
Pochgasse 31
7800 Freiburg
07 61 / 55 42 80

- Metzner Electronic
Postfach 11 14
7801 Umkirch
0 76 65 / 5817

- Software-Service
Ritterstr. 6
7833 Endingen
0 76 42 / 38 75

- Bürobedarf Friedrich Resin
Am Dreispitz
7852 Binzen
0 76 21 / 66 01-0

- Hettler GmbH
Lenzburger Str. 4
7890 Waldshut-Tiengen
0 77 51 / 30 94

- EDV + BÜRO SERVICE HESS
Ulrichstr. 24 B
7918 Illertissen
0 73 03 / 4 11 89

- Elektronik Center
Hindenburgstr. 45
8100 Garmisch-Partenkirchen
0 88 21 / 7 15 55

- Elektronik Center
Wachterstr. 13
8170 Bad Tölz
0 80 41 / 4 15 65

- FS Computer
Hechtseestr. 1
8208 Kolbermoor
0 80 31 / 9 46 14

- Computerzubehör/ PD-Service
Robert Rehrl
Stettener Weg 8
8221 Teisendorf
0 86 66 / 62 49

- Dutge Computer
Kantstr. 15
8225 Traunreut
0 86 69 / 3 67 53

- A.SPITZER EDV-Beratung
Sunklergäßchen 6
8240 Berchtesgaden
0 86 52 / 6 31 02

- Andreas Meyer Hard- und Software
Frauenhofer Str. 29
8255 Schwindegg

- Karstein Datentechnik
Aicha 10a
8451 Birgland
09 86 / 10 28

- AGP-Shop oHG
Auf der Schanze 4
8490 Cham
0 99 71 / 97 23

- hib Computer GmbH
Äußere Bayreuther Str. 57a-59
8500 Nürnberg 21
09 11 / 56 29 26

- Graf & Schick EDV-Lösungen
Hauptstr. 32 a
8542 Roth
0 91 71 / 50 58-59

- Schöll Computercenter
Dominikanerplatz 5
8700 Würzburg
09 31 / 41 90 60

- VC GmbH
Schwalbenstr. 7
8901 Stadtbergen
08 21 / 43 50 26